Investigating gradient descent in variational inference

4YP Number: 12140

Benjamin Gutteridge

New College

University of Oxford

Supervised by

Prof. Michael Osborne

Trinity Term 2021

Abstract

The aim of many machine learning applications is the modelling of data. We may wish to perform classification of images, clustering of users by preference for recommender systems, or regression for predicting financial markets; for all of these and limitless other applications, our goal is to learn a model from some given data. In some cases, we are content with so-called 'discriminative' models, those where we care only about learning the predictor function of interest. Generally though, the aim is to learn the true underlying model, the system that generated our data and that describes a real-world process or phenomenon. These are 'generative' models, as they not only allow us to learn systems and make predictions, but also to generate synthetic data similar to that upon which it was trained (Jebara 2012). We obtain such a generative model by performing Bayesian posterior inference: choose a distributional family to model the data, and learn the distribution of model parameters conditioned on that data.

The difficulty in practical inference applications is that in all but the most simple cases, we must solve an intractable integral. If we cannot evaluate the posterior analytically, we must resort to approximation, which can be broadly divided into two categories: stochastic methods such as Monte Carlo, which guarantee convergence but are often slow and inefficient, and deterministic methods (Bishop 2006). This report focuses on the deterministic approach, also called variational inference (VI; Jordan et al. 1998; Wainwright and Jordan 2007), which frames posterior inference as an optimisation problem, and often approximates the posterior much more quickly and efficiently than by random sampling methods.

As an optimisation problem, we can approach VI with specialised algorithms such as coordinate ascent variational inference (Bishop 2006), or general-purpose algorithms like stochastic gradient descent (Robbins and Monro 1951) – in either case, we will find gradients, their calculation and estimation, at the centre of the method.

In this report we provide a review of VI, with a particular focus on gradient descent algorithms, estimating accurate gradients, and how this affects the accuracy of posterior approximation and convergence rates. We also consider Bayesian quadrature (probabilistic numerical integration), its current applications to VI and possible novel uses, though experimentation in this area was unfortunately limited due to time constraints and negative results.

Acknowledgements

I would like to thank Edward Wagstaff for helpful discussions and support throughout this project, as well as several hugely productive pair coding sessions.

I am also grateful to my supervisor, Professor Michael Osborne, for his guidance throughout my 4YP; for his support with graduate applications; and for his encouragement and patience.

Contents

1 Introduction

The aim of many machine learning applications is the modelling of data. We may wish to perform classification of images, clustering of users by preference for recommender systems, or regression for predicting financial markets; for all of these and limitless other applications, our goal is to learn a model from some given data. In some cases, we are content with so-called 'discriminative' models, those where we care only about learning the predictor function of interest. Generally though, the aim is to learn the true underlying model, the system that generated our data and that describes a real-world process or phenomenon. These are 'generative' models, as they not only allow us to learn systems and make predictions, but also to generate synthetic data similar to that upon which it was trained (Jebara 2012). Our generative model takes the form of a probability distribution, and will often be of a known or chosen family of distributions parameterised by some vector of model parameters θ. Fully defining our model, then, amounts to learning the most appropriate θ, given some data X. Maximum likelihood estimation offers one simple approach to this, by choosing model parameters $\hat{\theta}_{\mathrm{MLE}}$ that maximise likelihood $p(X|\theta)$, but this is only a point estimate and is therefore uninformative (Bishop 2006). A superior approach is posterior inference: learning the distribution of model parameters conditioned on data. We obtain the posterior using Bayes' theorem:

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)}, \tag{1.1}$$

where $p(X|\theta)$ is the model likelihood, $p(\theta)$ is the prior over model parameters, and $p(X)$ is our model evidence. As we know the distributional family of our generative model, we usually know the likelihood for given θ, and are able to evaluate it. The prior, which encodes any knowledge or beliefs we may have about the model parameters, is chosen by us. This leaves the model evidence, also known as marginal likelihood, which is effectively a weighting of likelihoods over all possible θ:

$$p(X) = \int p(X|\theta)p(\theta)d\theta. \tag{1.2}$$

The difficulty in practical inference applications is that in all but the most simple cases, this integral is intractable, meaning that we cannot normalise the posterior. If we cannot evaluate the posterior analytically, we must resort to approximation, which can be broadly divided into two methods:

stochastic and deterministic (Bishop 2006). Stochastic methods use sampling-based numerical integration to approximate the intractable evidence, typically using Monte Carlo integration, or some variant of this such as Markov chain Monte Carlo (Hastings 1970; see Section 7.2.1). Monte Carlo integration asymptotically converges to the true integral for sufficiently many samples, but is often too slow for large and/or high dimensional datasets, and is inefficient in its use of data (O'Hagan 1987). Conversely, deterministic approximate inference, also called variational inference (VI; Jordan et al. 1998; Wainwright and Jordan 2007), frames posterior inference as an optimisation problem, and allows the model evidence and posterior to be approximated simultaneously, much more quickly and efficiently than by random sampling methods.

As an optimisation problem, we can approach VI with specialised algorithms such as coordinate ascent variational inference (Bishop 2006), or general-purpose algorithms like stochastic gradient descent (Robbins and Monro 1951) – in either case we will find gradients, their calculation and estimation, at the centre of the method.

In this report we provide a review of VI, with a particular focus on gradient descent (GD) algorithms, estimating accurate gradients, and how this affects the accuracy of posterior approximation and convergence rates. Throughout, we will focus our analysis on the problem of point clustering, a foundational unsupervised learning problem with applications across a wide range of fields, using a Bayesian Gaussian mixture model (GMM; Ghahramani, Beal, et al. 1999). We note that this is primarily a demonstrative example; while used broadly, GMMs are a relatively simple application of VI, with many of the key equations and expectations being available in closed form. We do not have this luxury for many real-world and state-of-the-art applications of VI, such as variational autoencoders (Kingma and Welling 2014), and so we also consider a general treatment of gradient estimation for stochastic VI.

Section 2 provides an overview of variational inference, deriving the variational objective function to optimise, and introduces the coordinate ascent and gradient descent algorithms. Section 3 introduces the variational mixture of Gaussians and derives coordinate ascent variational inference updates for this problem, as well as gradients. Section 4 introduces the idea of natural gradients and 'stochastic variational inference' (Hoffman et al. 2013), an update to classical VI for big data applications that utilises mini-batching to estimate gradients. Section 5 broadens our consideration of stochastic gradient descent VI to consider two widely used gradient estimators,

score function and pathwise, as well as briefly looking at alternative variational objective functions that produce lower-variance gradient estimates. Section 6 details experimental results comparing the variational inference algorithms covered in this report. Lastly, Section 7 contains a theoretical overview of Bayesian quadrature, a probabilistic approach to numerical integration, reviews its uses in variational inference and considers potential areas for future research.

2 Variational Inference

In this section we derive the variational lower bound, our objective function we wish to maximise in variational inference. We then introduce two general VI algorithms: coordinate ascent variational inference and gradient descent.

2.1 The variational lower bound

We consider our standard Bayesian inference problem with a vector of datapoints X distributed by a model parameterised by a real vector of variables θ. We now introduce a new vector of latent variables Z, of equal size to X and representing some unknown information about each datapoint. Both θ and Z are unobserved variables and can be taken together, but for this report and our GMM example it is convenient to separate them. As the elements of Z scale with X and contain information about individual datapoints, we also call them 'local' variables, whereas θ is constant with the size of X and describes the behaviour of the overall model, and so is a vector of 'global' variables. Our posterior inference equation is therefore

$$p(Z, \theta|X) = \frac{p(X|Z, \theta)p(Z, \theta)}{\int_Z \int_\theta p(X|Z, \theta)p(Z, \theta)dZd\theta}, \tag{2.1}$$

and our goal is to approximate it.

2.2 The variational lower bound

The key idea of variational inference is to use a 'variational distribution', $q(Z, \theta)$, to approximate the true posterior $p(Z, \theta|X)$ that we would ultimately like to obtain. We can measure the accuracy

of this approximation using the Kullback-Leibler divergence (Kullback and Leibler 1951):

$$D_{\mathrm{KL}}(q(Z,\theta) \parallel p(Z,\theta|X)) = \int_Z \int_\theta q(Z,\theta) \ln\left(\frac{q(Z,\theta)}{p(Z,\theta|X)}\right) d\theta dZ. \tag{2.2}$$

This is a measure of how much one probability distribution varies from another, though it is worth noting that it is not a 'distance' between distributions, as KL divergence is not symmetric; i.e. $D_{\mathrm{KL}}(p \parallel q) \neq D_{\mathrm{KL}}(q \parallel p)$. Ideally the variational distribution will be identical to the posterior, in which case $D_{\mathrm{KL}}(q(Z,\theta) \parallel p(Z,\theta|X)) = 0$, but otherwise the divergence is always greater than zero. This would suggest that our goal is to minimise the KL divergence, but we cannot do this directly, as we would need to know the posterior. Fortunately, we can manipulate the KL divergence into a more cooperative form:

$$\begin{aligned} D_{\mathrm{KL}}(q(Z,\theta) \parallel p(Z,\theta|X)) &= \int_Z \int_\theta q(Z,\theta) \left(\ln q(Z,\theta) - \ln\left(\frac{p(Z,\theta,X)}{p(X)}\right)\right) d\theta dZ \\ &= \int_Z \int_\theta \left(-q(Z,\theta) \ln\left(\frac{p(X,Z,\theta)}{q(Z,\theta)}\right) + q(Z,\theta) \ln p(X)\right) d\theta dZ \\ &= -\mathcal{L}[q] + \ln p(X). \end{aligned} \tag{2.3}$$

We have related the KL divergence to the log-evidence via a difference term, $\mathcal{L}$, which we write above as a functional of the variational distribution q, but which can also be written as a function of unobserved variables, $\mathcal{L}(Z,\theta)$. We call this term the evidence lower bound (ELBO), as it is a lower bound on the log-evidence, a result we prove below:

$$\begin{aligned} \ln p(X) &= \ln\left(\int_Z \int_\theta p(X,Z,\theta) d\theta dZ\right) \\ &= \ln\left(\int_Z \int_\theta \frac{p(X,Z,\theta)}{q(Z,\theta)} q(Z,\theta) d\theta dZ\right) \\ &\geq \int_Z \int_\theta q(Z,\theta) \ln\left(\frac{p(X,Z,\theta)}{q(Z,\theta)}\right) d\theta dZ \\ &= \mathcal{L}[q]. \end{aligned} \tag{2.4}$$

We make use of Jensen's inequality, $\ln(\mathbb{E}_q[.]) \geq \mathbb{E}_q[\ln(.)]$, since $\ln$ is a concave function.

Now that we have shown that the log-evidence can be decomposed as the sum of the ELBO and KL divergence, we can use the fact that the log-evidence itself is independent of q to conclude the following: minimising the KL divergence between the variational and posterior distributions

is equivalent to maximising the ELBO. This is significant, as it means that, requiring knowledge only of the full joint distribution $p(X, Z, \theta)$, we can frame variational inference as an optimisation problem. By maximising the ELBO with respect to the variational distribution, we simultaneously obtain approximations of both the log-evidence and the posterior, the two quantities that we are chiefly concerned with finding in Bayesian inference.

This leaves the question of how to choose the variational distribution, which must be flexible enough to approximate the unknown posterior. Optimising over free form q is difficult, and so an effective approach is to choose a distributional family with a set of variational parameters ϕ, yielding $q_\phi = q(Z, \theta|\phi)$. Thus the ELBO becomes a function of ϕ and approximate inference simplifies to the following maximisation problem:

$$\arg\max_{\phi} \mathcal{L}(\phi). \tag{2.5}$$

We are now in a position to introduce two algorithms used in variational inference to perform this optimisation.

2.3 Coordinate ascent variational inference

Coordinate ascent variational inference (CAVI; Bishop 2006) is a VI algorithm that uses an iterative update scheme over variational parameters, similar to Gibbs sampling (S. Geman and D. Geman 1984). Consider our joint distribution $p(X, Z, \theta)$ of vectors of observed and local and global unobserved variables, and variational distribution $q(Z, \theta)$ (we omit the ϕ conditioning for now; variational parameters will appear during derivation). Making only the assumption that local and global variables are independent and we can factorise between them, so $q(Z, \theta) = q(Z)q(\theta)$, we manipulate the ELBO as to determine the optimal form of $q(Z)$:

$$\begin{aligned}
\mathcal{L}[q] &= \int_Z \int_\theta q(Z)q(\theta)(\ln p(X, Z, \theta) - \ln q(Z) - \ln q(\theta))d\theta dZ \\
&= \int_Z \left(q(Z) \left\{ \int_\theta q(\theta) \ln p(X, Z, \theta) d\theta \right\} - q(Z) \ln q(Z) \right) dZ + \int_\theta q(\theta) \ln q(\theta) d\theta \\
&= \int_Z \left(q(Z) \ln \tilde{p}(X, Z) - q(Z) \ln q(Z) \right) dZ + c \\
&= -D_{\mathrm{KL}}(q(Z) \parallel \tilde{p}(X, Z)) + c,
\end{aligned} \tag{2.6}$$

where $\ln \tilde{p}(X, Z) = \mathbb{E}_{q(\theta)}[\ln p(X, Z, \theta)]$ and c is a constant. We can see that maximising the ELBO is equivalent to minimising KL divergence between $\tilde{p}$ and $q(Z)$. This minimum is achieved by a divergence of zero, which occurs when $\tilde{p} = q(Z)$. We can therefore derive the optimal form of $q(Z)$ (denoted by *) as

$$q^*(Z) = \mathbb{E}_{q(\theta)}[\ln p(X, Z, \theta)]. \tag{2.7}$$

By the same approach, the optimal form of $q^*(\theta)$ is

$$q^*(\theta) = \mathbb{E}_{q(Z)}[\ln p(X, Z, \theta)]. \tag{2.8}$$

These are specific instances of the general CAVI update rule, which states that for any vector of unobserved variables $\mathbf{Z}$ whose variational distribution factorises as $q(\mathbf{Z}) = \prod_i^i q_i(Z_i)$:

$$\ln q_j^*(\mathbf{Z}_j) = \mathbb{E}_{i \neq j}[\ln p(X, \mathbf{Z})] + c. \tag{2.9}$$

This says that the optimal form of each variational factor is proportional to the expectation of the log-joint distribution taken over all other parameters. The constant term is a normalisation factor. As each q^* is approximating a posterior update, we can often guarantee the form of the approximation with sensible choice of conjugate priors, which will usually allow the constant term to be generated automatically.

Deriving the distributional form of each $q_j^*(\mathbf{Z}_j)$ allows us to find update equations for the variational parameters ϕ. CAVI is then simply a matter of updating each variational parameter iteratively until convergence is reached, i.e. when the ELBO reaches a chosen threshold. It is worth noting that the ELBO is generally nonconvex and therefore CAVI will only guarantee convergence to a local maximum, though this is often suitable or can be addressed by using multiple initialisations. In some cases, the optimal solution will be multimodal and one of these local optima is equal to the global optimum; the variational mixture of Gaussians is one such example (due to label switching). We derive the CAVI algorithm for a GMM clustering problem in Section 3.2.

2.4 Gradient descent

Gradient descent is an iterative optimisation algorithm in which a local minimum of a cost function is found by taking steps down the gradient of the function. In VI we are trying to maximise the ELBO with respect to variational parameters ϕ, and can do so with the following update equation:

$$\phi_{t+1} = \phi_t + \rho_t \nabla_{\phi_t} \mathcal{L}. \tag{2.10}$$

Note that, as a maximisation problem, this is technically gradient *ascent*, but we use the term gradient descent by convention. In some cases the gradient can be computed analytically; we derive gradients for the variational mixture of Gaussians in Section 3.3, and demonstrate the relationship between CAVI and gradient descent. In most cases though, the gradient is either intractable or computationally impractical to obtain due to the size of the dataset used. In these cases we use stochastic gradient descent (SGD; Robbins and Monro 1951). Stochastic gradient descent relies on finding noisy but unbiased estimates of the gradient during each iteration, and provably converges to a (local, if nonconvex) minimum when using an appropriate step size schedule Spall 2003. We look at two applications of SGD to VI in this report, both of which are sampling approaches based on Monte Carlo integration:

$$\langle f \rangle = \frac{1}{L} \sum_{l=1}^{L} f(\hat{x}_l) \approx \mathbb{E}_{p(x)}[f(x)] = \int f(x) p(x) dx \tag{2.11}$$

where $\hat{x}_l \sim p(x)$. The first uses mini-batch estimates of the dataset X and is detailed in Section 4. The second uses samples of global variables θ and is based on general gradient estimation methods which we detail in Section 5. Note that we use $\langle . \rangle$ notation throughout this report to denote estimates of expectations obtained via Monte Carlo methods.

3 Variational Gaussian Mixture Model

We motivate our exploration of variational inference with the example of a fixed-covariance multivariate Bayesian Gaussian mixture model (GMM; Ghahramani, Beal, et al. 1999) point clustering problem.

Clustering is the automatic grouping of datapoints by similarity, and is a foundational unsupervised learning problem of data analysis with endless applications. Examples include within finance, in the identification of market sectors (Arnott 1980); the distinguishing of features in image and natural language processing (Di Marco and Navigli 2013; Filipovych et al. 2011), and in recommender systems for grouping users with similar interests (Karlgren 1994). GMM fitting is especially powerful in clustering as it produces a generative model, and variational methods allow for fast, efficient Bayesian inference of such models. We begin by defining the full joint and variational distributional families parameterised by ϕ, and then use Equation 2.9 to derive the optimal variational distributions and CAVI update equations for ϕ. We then consider the gradient descent algorithm and derive gradients of the ELBO with respect to ϕ.

A GMM is a linear superposition of weighted Gaussians:

$$p(x_n|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k). \tag{3.1}$$

We define a D-dimensional dataset $X = \{x_1, x_2, ...x_N\}$ with corresponding latent variables $Z = \{z_1, z_2, ...z_N\}$, where each z_n is a one-hot vector of length K indicating which of K mixture components generated point x_n. The mixture model is parameterised by $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$, where π_k denotes mixture weight, μ_k denotes Gaussian mean and Σ_k denotes covariance of mixture component k.

For a GMM, CAVI updates are similar to the Expectation-Maximisation algorithm (Dempster et al. 1977; Neal and Hinton 1998). Broadly, each iteration consists of:

1. An 'expectation' step, where predictions for the latent variables Z (i.e. point assignment to mixture components) are made based on current model parameter estimates.

2. A 'maximisation' step, where model parameters are updated according to update equations, one at a time, each update guaranteeing an increase in the ELBO.

These two steps are repeated until convergence. In this section we derive the CAVI algorithm and update equations for a Bayesian GMM with fixed covariances $\Sigma_k = \Sigma$. We derive the joint and variational distributional forms, as well as the ELBO gradients with respect to variational parameters to demonstrate the link between gradients and CAVI. We will use this GMM and the results of this

section throughout the report as much of the theory and derivations are closely linked to stochastic approaches.

3.1 The joint distribution

The full joint distribution of p is as follows, dependencies given by the graphical model in Figure 3.1:

$$p(X, Z, \theta) = p(X, Z, \pi, \mu, \Sigma) = p(X|Z, \mu, \Sigma)p(Z|\pi)p(\pi)p(\mu). \tag{3.2}$$

Let us consider each term. The first, the model likelihood, is simply a product of multivariate

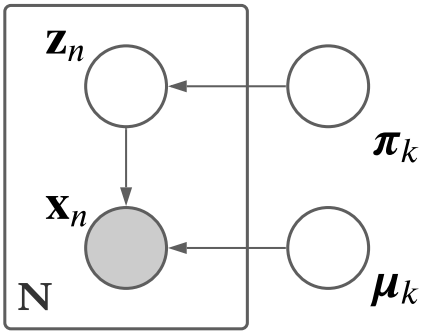

Figure 3.1: Graphical model showing the relationships between variables in a GMM. The shaded node indicates that x_n is observed.

Gaussians, one for each datapoint (as z_{nk} is a one-hot vector):

$$p(X|Z, \mu, \Sigma) = \prod_{n=1}^{N} \prod_{k=1}^{K} \mathcal{N}(x_n|\mu_k, \Sigma)^{z_{nk}}. \tag{3.3}$$

For a single point and its known cluster assignment z_{nk},

$$p(x_n|z_{nk} = 1, \mu, \Sigma) = \mathcal{N}(x_n|\mu_k, \Sigma). \tag{3.4}$$

The conditional distribution of Z is a multinomial distribution over the mixture weights, where $\sum_{k=1}^{K} \pi_k = 1$:

$$p(Z|\pi) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}} \quad p(z_{nk} = 1) = \pi_k. \tag{3.5}$$

Next we consider the prior distributions on the model parameters $p(\pi)$ and $p(\mu)$, using conjugate priors for the two likelihoods $p(X|Z,\mu)$ and $p(Z|\pi)$. Using conjugate priors guarantees the distributional form of the posteriors and makes derivations much simpler. For given z_{nk}, the likelihood for μ is a multivariate Gaussian with known covariance (Equation 3.4), for which the conjugate prior is also Gaussian, hence we define the prior over μ as

$$p(\mu_k) = \mathcal{N}(\mu_k|m_0, C_0), \tag{3.6}$$

where means μ_k are independent of one another, and the prior factorises into $p(\mu) = \prod_{k=1}^{K} p(\mu_k)$.

The likelihood for π, Equation 3.5, is a multinomial, for which the conjugate prior is a Dirichlet distribution. Hence we define the prior over π as

$$p(\pi) = \mathrm{Dir}(\pi|\alpha_0). \tag{3.7}$$

It is worth briefly looking at the Dirichlet distribution and the significance of its parameters.

3.1.1 Dirichlet distribution

The form of the Dirichlet distribution is as follows:

$$\mathrm{Dir}(\pi|\alpha) = C(\alpha) \prod_{k=1}^{K} \pi_k^{\alpha_k - 1}, \tag{3.8}$$

where

$$C(\alpha) = \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma(\alpha_k)}, \tag{3.9}$$

for Gamma function $\Gamma(.)$. This is a multivariate probability distribution with a set of K concentration parameters $\alpha = [\alpha_1, \alpha_2, ..., \alpha_K]^T$. For weights in a mixture model, the concentration parameter provides two intuitive pieces of information: the relative significance of each component, and their sparsity. For example, uniform $\alpha_k = \alpha_0$ means we have no preference between mixture components; $\mathbb{E}_\alpha[\pi_k] = \alpha_k / \sum_{j=1}^{K} \alpha_j = 1/K$. If α_0 is large, sampled π_k will be reasonably similar and tend to $1/K$ as $\alpha_0 \to \infty$. If α_0 is small, $p(\pi)$ favours a sparse mixture with one (random) π_k close to one and all other $\pi_{j \neq k} \to 0$.

This is a significant advantage of the posterior inference approach to GMM clustering, as we need not guess the number of clusters. We can simply choose a reasonably large K and the model will eliminate unnecessary components automatically, by driving those values of α_k to zero.

3.2 The variational distribution

Next, we consider the form of $q(Z, \pi, \mu)$, our variational approximation to the posterior for the GMM. We make only one assumption: that we can factorise between the latent variables and the model parameters, known as the mean-field approximation:

$$p(Z, \pi, \mu|X) \approx q(Z, \pi, \mu) = q(Z)q(\pi, \mu). \tag{3.10}$$

Now we can derive the optimal distributions $q^*(Z)$ and $q^*(\pi, \mu)$ using Equation 2.9.

Let us first consider the optimal variational distribution of local variables $q^*(Z)$. We decompose the log-joint, absorbing terms that are not dependent on Z into a constant c, then substitute in known distributions:

$$\begin{aligned}
\ln q^*(Z) &= \mathbb{E}_\mu[\ln p(X|Z, \mu)] + \mathbb{E}_\pi[\ln p(Z|\pi)] + c \\
&= -\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\mathbb{E}_{\mu_k}[(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k)] + \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\mathbb{E}_{\pi_k}[\ln \pi_k] + c \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\rho_{nk} + c,
\end{aligned} \tag{3.11}$$

where

$$\rho_{nk} = -\frac{1}{2}\mathbb{E}_{\mu_k}[(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k)] + \mathbb{E}_{\pi_k}[\ln \pi_k]. \tag{3.12}$$

Exponentiating both sides, and normalising,

$$\begin{aligned}
q^*(Z) &\propto \prod_{n=1}^{N}\prod_{k=1}^{K} \rho_{nk}^{z_{nk}} \\
q^*(Z) &= \prod_{n=1}^{N}\prod_{k=1}^{K} r_{nk}^{z_{nk}},
\end{aligned} \tag{3.13}$$

where $r_{nk} = \rho_{nk} / \sum_{j=1}^{K} \rho_{nj}$. So the variational parameter of Z element z_{nk} is r_{nk}, the responsibility

borne by the kth mixture component for datapoint x_n. We can see that $q^*(Z)$ is a multinomial distribution, the same form as the mixing coefficient prior $p(\pi)$.

At this point it is sensible to derive three useful statistics of the responsibilities:

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{3.14}$$

$$\bar{x}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} x_n \tag{3.15}$$

$$S_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (x_n - \bar{x}_k)^T (x_n - \bar{x}_k). \tag{3.16}$$

These are analogous to the statistics of a dataset of size N with mean $\bar{x}$ and covariance S, with each point weighted by the responsibility of mixture component k.

Next we consider the optimal variational family of global variables, $q^*(\pi, \mu)$. Decomposing the log-joint, we see that $q^*(\pi, \mu)$ can be split into terms only involving π and those only involving μ, so we can factorise a step further:

$$\begin{aligned} \ln q^*(\pi, \mu) &= \mathbb{E}_Z[\ln p(X|Z, \mu) + \ln p(\mu)] + \mathbb{E}_Z[\ln p(Z|\pi) + \ln p(\pi)] + c \\ &= \ln q^*(\pi) + \ln q^*(\mu). \end{aligned} \tag{3.17}$$

Note that this is not an assumption but comes from our joint distribution. We derive the optimal variational family of $q(\pi)$ using a similar procedure to $q(Z)$:

$$\begin{aligned} \ln q^*(\pi) &= \mathbb{E}_Z[\ln p(Z|\pi) + \ln p(\pi)] + c \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{E}_Z[z_{nk}] \ln \pi_k + (\alpha_0 - 1) \sum_{k=1}^{K} \ln \pi_k + c \\ &= \sum_{k=1}^{K} \left(\sum_{n=1}^{N} r_{nk} \right) \ln \pi_k + (\alpha_0 - 1) \sum_{k=1}^{K} \ln \pi_k + c \\ &= (N_k + \alpha_0 - 1) \sum_{k=1}^{K} \ln \pi_k + c \\ q^*(\pi|\alpha) &= c \prod_{k=1}^{K} \pi_k^{N_k + \alpha_0 - 1}. \end{aligned} \tag{3.18}$$

This is equal to the form of Equation 3.8 when $c = C(\alpha)$ (which it must be to normalise the

distribution), and thus we find that $q^*(\pi|\alpha) = \mathrm{Dir}(\pi|\alpha)$ where $\alpha_k = N_k + \alpha_0$. As expected, our approximate posterior for π is the same as the conjugate prior distribution we have chosen.

Lastly, the optimal variational family of $q(\mu)$:

$$\begin{aligned}
\ln q^*(\mu) &= \mathbb{E}_Z[\ln p(X|Z,\mu) + \ln p(\mu)] + c \\
&= -\frac{1}{2}\left(\sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}_Z[z_{nk}](x_n-\mu_k)^T\Sigma^{-1}(x_n-\mu_k) + \sum^{K}(\mu_k-m_0)^T C_0^{-1}(\mu_k-m_0)\right) + c, \\
\ln q^*(\mu_k) &= -\frac{1}{2}\left(\sum_{n=1}^{N} r_{nk}(x_n-\mu_k)^T\Sigma^{-1}(x_n-\mu_k) + (\mu_k-m_0)^T C_0^{-1}(\mu_k-m_0)\right) + c.
\end{aligned} \tag{3.19}$$

The log distribution is a sum over K and thus μ factorises into individual μ_k, as $p(\mu)$ does. By conjugacy, $q^*(\mu_k)$ will be a Gaussian distribution, which we parameterise by mean and covariance m_k, C_k:

$$q^*(\mu_k) = \mathcal{N}(\mu_k|m_k, C_k). \tag{3.20}$$

Ignoring all terms outside the Gaussian exponent for now, we can use the following expression for the product of Gaussian densities (Petersen, Pedersen, et al. 2008) to obtain the product of $N+1$ Gaussians:

$$\begin{gathered}
\mathcal{N}(m_1,\Sigma_1)\cdot\mathcal{N}(m_2,\Sigma_2) \propto \mathcal{N}(m_c,\Sigma_c), \\
m_c = (\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}m_1+\Sigma_2^{-1}m_2), \quad \Sigma_c = (\Sigma_1^{-1}+\Sigma_2^{-1})^{-1},
\end{gathered} \tag{3.21}$$

and obtain

$$\begin{aligned}
m_k &= \left(\sum_{n=1}^{N} r_{nk}\Sigma^{-1} + C_0^{-1}\right)^{-1}\left(\sum_{n=1}^{N}\Sigma^{-1}r_{nk}x_n + C_0^{-1}m_0\right) \\
&= \left(N_k\Sigma^{-1} + C_0^{-1}\right)^{-1}\left(N_k\Sigma^{-1}\bar{x}_k + C_0^{-1}m_0\right), \\
C_k &= \left(N_k\Sigma^{-1} + C_0^{-1}\right)^{-1}.
\end{aligned} \tag{3.22}$$

We have now derived the forms of the factorised variational distribution and the update equations for our responsibilities r_{nk} and variational parameters $\phi = \{\alpha, m, C\}$. Our collected CAVI updates

are:

$$r_{nk} = \frac{\rho_{nk}}{\sum_{j=1}^{K} \rho_{nj}}, \tag{3.23}$$

$$\alpha_k = \alpha_0 + N_k, \tag{3.24}$$

$$m_k = \left(N_k \Sigma^{-1} + C_0^{-1}\right)^{-1} \left(N_k \Sigma^{-1} \bar{x}_k + C_0^{-1} m_0\right), \tag{3.25}$$

$$C_k = \left(N_k \Sigma^{-1} + C_0^{-1}\right)^{-1}. \tag{3.26}$$

We summarise CAVI in Algorithm 1; it is extremely powerful, often nearing convergence in only a few iterations; see Figure 3.2.

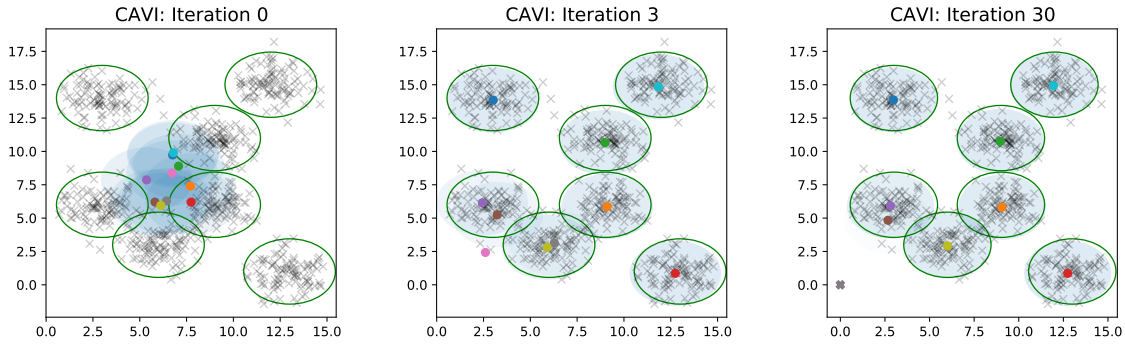

Figure 3.2: Plotting a Gaussian mixture model using expected values of π, μ after n iterations on a synthetic dataset. More information and experiments are in Section 6

3.3 ELBO gradients

In order to perform gradient descent updates, we must be able to calculate the gradient of the ELBO with respect to our variational parameters. As r, the variational parameter for the local variables Z, is discrete, we maintain the 'expectation' style step for its calculation derived in Section 3.2 rather than using gradient descent steps (this method is also used in the stochastic variational inference algorithm of Hoffman et al. 2013, see Section 4). To derive the gradients of the ELBO with respect to α, m, C, the variational parameters of global variables π, μ, we begin with the full

Algorithm 1: Coordinate ascent variational inference for a fixed covariance GMM

Input: Data X, number of components K, priors α_0, m_0, C_0, fixed Gaussian covariance Σ, step size schedule $\rho(t)$

Output: Variational distributions $q(z_n|r_n), q(\pi|\alpha), q(\mu_k|m_k, C_k)$

Initialise: $t = 0$, variational parameters $r = r_{1:N,1:K}, \alpha = \alpha_{1:K}, m = m_{1:K}, C = C_{1:K}$

while *the* ELBO *has not converged* **do**

  for $n \in \{1, ..., N\}, \quad k \in \{1, ..., K\}$ **do**

    Set $\rho_{nk} \leftarrow -\frac{1}{2}\mathbb{E}_{\mu_k}[(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k)] + \mathbb{E}_{\pi_k}[\ln \pi_k]$;

  end

  for $n \in \{1, ..., N\}, \quad k \in \{1, ..., K\}$ **do**

    Set $r_{nk} \leftarrow \rho_{nk} / \sum_{j=1}^{K} \rho_{nj}$

  end

  for $k \in \{1, ..., K\}$ **do**

    Set $\alpha_k \leftarrow \alpha_0 + N_k$;

    Set $m_k \leftarrow \left(N_k\Sigma^{-1} + C_0^{-1}\right)^{-1}\left(N_k\Sigma^{-1}\bar{x}_k + C_0^{-1}m_0\right)$;

    Set $C_k \leftarrow \left(N_k\Sigma^{-1} + C_0^{-1}\right)^{-1}$;

  end

  Compute the ELBO(α, m, C, r) using Equation 3.27;

  $t \leftarrow t + 1$;

end

ELBO expression below:

$$\begin{aligned}\mathcal{L}[q] = &\int_Z \int_\mu \int_\pi q(Z, \pi, \mu) \ln\left(\frac{p(X, Z, \pi, \mu)}{q(Z, \pi, \mu)}\right) d\pi d\mu dZ \\ =&\mathbb{E}[\ln p(X, Z, \pi, \mu)] - \mathbb{E}[\ln q(Z, \pi, \mu)] \\ =&\mathbb{E}[\ln p(X|Z, \mu)] + \mathbb{E}[\ln p(Z|\pi)] + \mathbb{E}[\ln p(\pi)] + \mathbb{E}[\ln p(\mu)] \\ &- \mathbb{E}[\ln q(Z)] - \mathbb{E}[\ln q(\pi)] - \mathbb{E}[\ln q(\mu)].\end{aligned} \tag{3.27}$$

Note that we omit subscripts, but each expectation is taken over Z, π, μ. Considering each component:

$$\mathbb{E}_{Z,\mu}[\ln p(X|Z,\mu)] = -\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}_Z[z_{nk}]\left(\mathbb{E}_{\mu_k}[(x_n-\mu_k)^T\Sigma^{-1}(x_n-\mu_k)] + D\ln(2\pi) + \ln|\Sigma|\right), \tag{3.28}$$

$$\mathbb{E}_{Z,\pi}[\ln p(Z|\pi)] = \sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}_Z[z_{nk}]\mathbb{E}_{\pi_k}[\ln\pi_k], \tag{3.29}$$

$$\mathbb{E}_{\pi}[\ln p(\pi)] = \ln C(\alpha_0) + (\alpha_0-1)\sum_{k=1}^{K}\mathbb{E}_{\pi_k}[\ln\pi_k], \tag{3.30}$$

$$\mathbb{E}_{\mu}[\ln p(\mu)] = -\frac{1}{2}\sum_{k=1}^{K}\left(\mathbb{E}_{\mu_k}[(\mu_k-m_0)^TC_0^{-1}(\mu_k-m_0)] + D\ln(2\pi) + \ln|C_0|\right), \tag{3.31}$$

$$\mathbb{E}_{Z}[\ln q(Z)] = \sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}_Z[z_{nk}]\ln r_{nk}, \tag{3.32}$$

$$\mathbb{E}_{\pi}[\ln q(\pi)] = \ln C(\alpha) + \sum_{k=1}^{K}(\alpha_k-1)\mathbb{E}_{\pi_k}[\ln\pi_k], \tag{3.33}$$

$$\mathbb{E}_{\mu}[\ln q(\mu)] = -\frac{1}{2}\sum_{k=1}^{K}\left(\mathbb{E}_{\mu_k}[(\mu_k-m_k)^TC_k^{-1}(\mu_k-m_k)] + D\ln(2\pi) + \ln|C_k|\right). \tag{3.34}$$

We can use some standard properties of the categorical (a single trial multinomial), Dirichlet and Gaussian distributions, denoting entropy $\mathbb{E}_\mu[\ln q(\mu)]$ by $\mathcal{H}$ and $\hat{\alpha} = \sum_{k=1}^{K}\alpha_k$:

$$z_{nk} \sim \text{Mult}(1, r_{nk}), \quad \mathbb{E}[z_{nk}] = r_{nk}; \tag{3.35}$$

$$\pi_k \sim \text{Dir}(\alpha), \quad \mathbb{E}[\ln\pi_k] = \psi(\alpha_k) - \psi(\hat{\alpha}); \tag{3.36}$$

$$\mu_k \sim \mathcal{N}(m_k, C_k), \quad \mathcal{H}[q(\mu_k)] = \frac{D}{2}\ln(2\pi e) + \frac{1}{2}\ln|C_k|; \tag{3.37}$$

where ψ is the digamma function. We can use the standard evaluation of the expectations in Equations 3.28 and 3.31 (Petersen, Pedersen, et al. 2008):

$$\mathbb{E}_{\mu_k}[(\mu_k-x_n)^T\Sigma^{-1}(\mu_k-x_n)] = (m_k-x_n)^T\Sigma^{-1}(m_k-x_n) + \text{Tr}(\Sigma^{-1}C_k), \tag{3.38}$$

$$\mathbb{E}_{\mu_k}[(\mu_k-m_0)^TC_0^{-1}(\mu_k-m_0)] = (m_k-m_0)^TC_0^{-1}(m_k-m_0) + \text{Tr}(C_0^{-1}C_k). \tag{3.39}$$

It is sensible to further evaluate Equation 3.28, focusing only on terms in m_k and absorbing all

others into a constant, and using the fact that $x^T A x = \mathrm{Tr}(A x x^T)$ and Equations 3.14, 3.15, and 3.16:

$$\begin{aligned}
&\sum_{n=1}^{N} r_{nk}(m_k - x_n)^T \Sigma^{-1}(m_k - x_n) \\
&= \sum_{n=1}^{N} r_{nk}\left((m_k - \bar{x}_k - x_n + \bar{x}_k)^T \Sigma^{-1}(m_k - \bar{x}_k - x_n + \bar{x}_k)\right) \\
&= \sum_{n=1}^{N} r_{nk}\left((m_k - \bar{x}_k)^T \Sigma^{-1}(m_k - \bar{x}_k) + (\bar{x}_k - x_n)^T \Sigma^{-1}(\bar{x}_k - x_n)\right. \\
&\quad \left. + (m_k - \bar{x}_k)^T \Sigma^{-1}(\bar{x}_k - x_n) + (\bar{x}_k - x_n)^T \Sigma^{-1}(m_k - \bar{x}_k)\right) \\
&= N_k(m_k - \bar{x}_k)^T \Sigma^{-1}(m_k - \bar{x}_k) \\
&\quad + \mathrm{Tr}\left(\Sigma^{-1}\left(S_k(m_k - \bar{x}_k)(N_k\bar{x}_k - N_k\bar{x}_k)^T + (N_k\bar{x}_k - N_k\bar{x}_k)(m_k - \bar{x}_k)^T\right)\right) \\
&= N_k(m_k - \bar{x}_k)^T \Sigma^{-1}(m_k - \bar{x}_k) + \mathrm{Tr}\left(\Sigma^{-1} S_k\right).
\end{aligned} \tag{3.40}$$

Finally we can calculate gradients with respect to α, m, C:

$$\begin{aligned}
\nabla_{\alpha_k}\mathcal{L} &= \nabla_{\alpha_k}\left(\mathbb{E}_{Z,\pi}[\ln p(Z|\pi)] + \mathbb{E}_{\pi}[\ln p(\pi)]] - \mathbb{E}_{\pi}[\ln q(\pi)]\right) \\
&= \left(\sum_{n=1}^{N} r_{nk} + (\alpha_0 - 1) - (\alpha_k - 1)\right)\nabla_{\alpha_k}\mathbb{E}_{\pi_k}[\ln \pi_k] - \mathbb{E}_{\pi_k}[\ln \pi_k] - \nabla_{\alpha_k}\ln C(\alpha) \\
&= (N_k + \alpha_0 - \alpha_k)(\psi'(\alpha_k) - \psi'(\hat{\alpha})) - (\psi(\alpha_k) - \psi(\hat{\alpha})) - (\psi(\hat{\alpha}) - \psi(\alpha_k)) \\
&= (N_k + \alpha_0 - \alpha_k)(\psi'(\alpha_k) - \psi'(\hat{\alpha})),
\end{aligned} \tag{3.41}$$

$$\begin{aligned}
\nabla_{m_k}\mathcal{L} &= \nabla_{m_k}\left(\mathbb{E}_{Z,\mu_k}[\ln p(X|Z,\mu_k)] + \mathbb{E}_{\mu_k}[\ln p(\mu_k)]\right) \\
&= \nabla_{m_k}\left(-\frac{1}{2}\sum_{n=1}^{N} r_{nk}(m_k - x_n)^T\Sigma^{-1}(m_k - x_n) - \frac{1}{2}(m_k - m_0)^T C_0^{-1}(m_k - m_0)\right) \\
&= -N_k\Sigma^{-1}(m_k - \bar{x}_k) - C_0^{-1}(m_k - m_0),
\end{aligned} \tag{3.42}$$

$$\begin{aligned}
\nabla_{C_k}\mathcal{L} &= \nabla_{C_k}\left(\mathbb{E}_{Z,\mu_k}[\ln p(X|Z,\mu_k)] + \mathbb{E}_{\mu_k}[\ln p(\mu_k)] + \mathbb{E}_{\mu_k}[\ln q(\mu_k)]\right) \\
&= \nabla_{C_k}\left(-\frac{1}{2}\sum_{n=1}^{N} r_{nk}\mathrm{Tr}(\Sigma^{-1}C_k) - \frac{1}{2}\mathrm{Tr}(C_0^{-1}C_k) + \frac{1}{2}\ln|C_k|\right) \\
&= \frac{1}{2}\left(C_k^{-1} - N_k\Sigma^{-1} - C_0^{-1}\right).
\end{aligned} \tag{3.43}$$

It is evident from these three gradients that the CAVI update equations (3.24, 3.25, 3.26) set

the gradient of the ELBO with respect to each variational parameter to zero, effectively making them perfect gradient descent 'leaps' to the maximum ELBO for each variational parameter with all others fixed. It should be stressed then that for the GMM a GD algorithm is an inferior version of CAVI and we would never use it in practice; we have derived it here for two reasons. Firstly, the stochastic variational algorithms for the GMM introduced in Section 4, based on mini-batch SGD, build on the GD algorithm and the derived gradients. Secondly, because while true GD is rarely used in practice, SGD very commonly is, and we are interested in the comparison of true gradients and noisy gradient estimates in GD algorithms – SGD is effectively trying to approximate true GD as closely as possible while reducing computation. We investigate this in Section 6.

Algorithm 2: Gradient descent variational inference for a fixed covariance GMM

Input: Data X, number of components K, priors α_0, m_0, C_0, fixed Gaussian covariance Σ, step size schedule $\rho(t)$
Output: Variational distributions $q(z_n|r_n), q(\pi|\alpha), q(\mu_k|m_k, C_k)$
Initialise: $t = 0$, variational parameters $r = r_{1:N,1:K}, \alpha = \alpha_{1:K}, m = m_{1:K}, C = C_{1:K}$
while *the* ELBO *has not converged* **do**
  Calculate matrix of responsibilities r following procedure of Algorithm 1;
  for $k \in \{1, ..., K\}$ **do**
    Set $\alpha_k \leftarrow \alpha_k + \rho(t)\nabla_{\alpha_k}\mathcal{L}$ using Equation 3.41;
    Set $m_k \leftarrow m_k + \rho(t)\nabla_{m_k}\mathcal{L}$ using Equation 3.42;
    Set $C_k \leftarrow C_k + \rho(t)\nabla_{C_k}\mathcal{L}$ using Equation 3.43;
  end
  Compute the ELBO(α, m, C, r) using Equation 3.27;
  $t \leftarrow t + 1$;
end

4 Stochastic Variational Inference

In this section, we address issues with the CAVI and GD algorithms we have already introduced, based on the stochastic variational inference (SVI) approach of Hoffman et al. 2013. Firstly, both CAVI and GD rely on calculating an $N \times K$ matrix of responsibilities in every iteration, a computationally expensive procedure, especially when many real-world applications use huge datasets. Secondly, we have already shown that the GD algorithm is inferior to CAVI; we can further show that gradients in the traditional Euclidean sense may be inappropriate for many VI problems, and suggest an alternative.

4.1 Mini-batch stochastic gradient descent

We have previously considered two methods of variational inference, CAVI and gradient descent. A downside of both methods is that they require analysing every point in the dataset for a single iteration. For large datasets, this is prohibitively computationally expensive, and especially wasteful considering that an initial update is made over the entire dataset using a completely uninformative random initialisation. A reasonable assumption would be that a random subset of the large dataset $S \subset X$, or even a single datapoint, would be sufficiently informative to generate a noisy but unbiased estimate of the gradient and allow SGD steps to be taken. We derive these estimators below for our GMM problem, and in Section 4.2.3 apply this same idea to an alternative gradient formulation.

4.1.1 Stochastic gradient estimates of the ELBO for the GMM

Recall that the unobserved variables of our GMM problem are both global model parameters π, μ and local latent variables Z. From Equation 3.23 we see that updating the responsibilities, $\{r_{nk}\}_{k=1}^{K}$, corresponding to local variable z_n relies only on the global variables π, μ, and on the nth datapoint x_n. Therefore, stochastic updates for each z_n are identical to their batch form, and we reduce computation by simply updating the responsibilities for only $S < N$ sample points.

To derive mini-batch gradient estimates of the ELBO with respect to α, m, C, we need Monte Carlo estimators of the gradients given by Equations 3.41, 3.42 and 3.43, all of which are summations over N via N_k and $\bar{x}_k$. We can derive these estimators with a re-imagining of our GD algorithm in which the full dataset consists of a single sample $\hat{x}_n$ repeated N times. This produces responsibility-weighted statistics $N_k = Nr_{nk}, \bar{x}_k = \hat{x}_n, S_k = 0$, with which we can apply the usual gradient update steps. This method yields gradient estimates from a single sample per iteration; to improve results we can average single-sample estimates over randomly sampled mini-batches of size S in each iteration. With respect to variational parameters α, m, C, the mini-batch gradient

estimates of the ELBO are:

$$\langle \nabla_{\alpha_k} \mathcal{L} \rangle = \frac{1}{S} \sum_{n=1}^{S} (N r_{nk} + \alpha_0 - \alpha_k)(\psi'(\alpha_k) - \psi'(\hat{\alpha})), \tag{4.1}$$

$$\langle \nabla_{m_k} \mathcal{L} \rangle = \frac{1}{S} \sum_{n=1}^{S} -N r_{nk} \Sigma^{-1}(m_k - \hat{x}_n) - C_0^{-1}(m_k - m_0), \tag{4.2}$$

$$\langle \nabla_{C_k} \mathcal{L} \rangle = \frac{1}{S} \sum_{n=1}^{S} \frac{1}{2} \left(C_k^{-1} - N r_{nk} \Sigma^{-1} - C_0^{-1} \right). \tag{4.3}$$

We can use these estimators to perform stochastic gradient descent updates as detailed in Algorithm 3.

Algorithm 3: Stochastic gradient descent variational inference for a fixed covariance GMM

Input: Data X, number of components K, priors α_0, m_0, C_0, fixed Gaussian covariance Σ, step size schedule $\rho(t)$
Output: Variational distributions $q(z_n|r_n), q(\pi|\alpha), q(\mu_k|m_k, C_k)$
Initialise: $t = 0$, variational parameters $r = r_{1:N,1:K}, \alpha = \alpha_{1:K}, m = m_{1:K}, C = C_{1:K}$
while *the* ELBO *has not converged* **do**
  Sample X without replacement, generate subset $X_s \subset X$ of length $S < N$;
  for $n \in \{1, ..., S\}, \quad k \in \{1, ..., K\}$ **do**
    Set $\hat{\rho}_{nk} \leftarrow -\frac{1}{2}\mathbb{E}_{\mu_k}[(x_n - \mu_k)^T \Sigma^{-1}(x_n - \mu_k)] + \mathbb{E}_{\pi_k}[\ln \pi_k]$;
  end
  for $n \in \{1, ..., S\}, \quad k \in \{1, ..., K\}$ **do**
    Set $\hat{r}_{nk} \leftarrow \hat{\rho}_{nk} / \sum_{j=1}^{K} \hat{\rho}_{nj}$
  end
  for $k \in \{1, ..., K\}$ **do**
    Set $\alpha_k \leftarrow \alpha_k + \rho(t) \langle \nabla_{\alpha_k} \mathcal{L} \rangle$ using Equation 4.1;
    Set $m_k \leftarrow m_k + \rho(t) \langle \nabla_{m_k} \mathcal{L} \rangle$ using Equation 4.2;
    Set $C_k \leftarrow C_k + \rho(t) \langle \nabla_{C_k} \mathcal{L} \rangle$ using Equation 4.3;
  end
  Compute the ELBO(α, m, C, r);
  $t \leftarrow t + 1$;
end

4.2 Natural gradient descent

In traditional gradient descent, for a function $L(w)$ parameters are updated by taking small steps in a Euclidean D-dimensional parameter space $w \in R^D$ such that $w_{t+1} = w_t + dw$, for $|dw|^2 = \sum_{i=1}^{D} dw_i^2$. For updating parameters of distributions, however, steps in Euclidean space are not necessarily optimal, or even meaningful. Hoffman et al. 2013 uses as an example two

pairs of univariate Gaussians, $\mathcal{N}(x_1|0, 0.01), \mathcal{N}(x_1|0.1, 0.01), \mathcal{N}(x_2|0, 10000), \mathcal{N}(x_2|10, 10000)$. In the first pair the Euclidean distance between the means is only 0.1, but the two distributions barely overlap. In the second pair the distributions overlap almost completely, yet the Euclidean distance between the means is one hundred times greater. Evidently, the Euclidean distance is not the most appropriate measure of similarity; we should consider a metric that is better suited to probability distributions. Euclidean space is a particular case of Riemannian parameter space, in which $|dw|^2 = \sum_{i,j} g_{ij}(w) dw_i dw_j$, and the direction of steepest descent is equal to $G(w)^{-1}\nabla L(w)$, where G is the matrix of elements g_{ij}. This direction is defined as the natural gradient $\tilde{\nabla} L(w)$ (Amari 1998), and offers an alternative to traditional gradient descent for appropriate choice of G. How should we choose G? As we are considering the similarity of variational distributions $q_\phi(Z, \theta)$ with differing parameterisations ϕ, a sensible choice is the symmetrised KL divergence between distributions, $D_{\mathrm{KL}}(q_\phi \parallel q_{\phi'}) + D_{\mathrm{KL}}(q_{\phi'} \parallel q_\phi)$, for which G is the Fisher information matrix of q:

$$G(\phi) = \mathbb{E}_\phi[(\nabla_\phi \ln q_\phi)(\nabla_\theta \ln q_\phi)^T] \tag{4.4}$$

In order to obtain the Fisher information matrix, we consider a reformulation of the GMM problem using the exponential family of distributions.

4.2.1 The exponential family

The stochastic natural gradient approaches of Sato 2001 and Hoffman et al. 2013 are built on standard behaviour of distributions within the exponential family. Probability distributions over x with vector parameters θ in the exponential family take the form

$$p(x|\theta) = h(x) \exp(\eta(\theta) \cdot T(x) - A(\theta)), \tag{4.5}$$

with the following components:

- Base measure $h(x)$: defines distributional family, along with $T(x)$.
- Natural parameter $\eta(\theta)$: an alternative parameterisation of the distribution. The natural parameter space, for which $p(x|\theta)$ is finite, is always convex.

- Sufficient statistic $T(x)$: called thus because it is a statistic of obersved data that holds all possible information about θ.
- Log-normaliser $A(\theta)$: ensures that $p(x|\theta)$ integrates to 1.

Consider the three key distributions in our GMM variational factorisation: the categorical $q(z_n|r_n)$, Gaussian $q(\mu_k|m_k, C_k)$ and Dirichlet $q(\pi|\alpha)$. The exponential family components of each of these distributions is given in Table 1:

Distribution	θ	$\eta(\theta)$	$h(x)$	$T(x)$	$A(\eta)$
Gaussian	$\begin{bmatrix}\mu \\ \Sigma\end{bmatrix}$	$\begin{bmatrix}\Sigma^{-1}\mu \\ -\frac{1}{2}\Sigma^{-1}\end{bmatrix}$	$(2\pi)^{-k/2}$	$\begin{bmatrix}x \\ xx^T\end{bmatrix}$	$-\frac{1}{4}\eta_1^T\eta_2^{-1}\eta_1 - \frac{1}{2}\ln\lvert -2\eta_2\rvert$
Dirchlet	α	α	$1/\prod_{k=1}^K \pi_k$	$[\ln\pi_1, \ldots \ln\pi_K]^T$	$\sum_{i=1}^k \ln\Gamma(\eta_i) - \ln\Gamma\left(\sum_{i=1}^k \eta_i\right)$
Categorical	r_n	$\ln r_n$	1	$z_n \begin{cases} z_{nk} = 1 \\ z_{n\setminus k} = 0 \end{cases}$	0

Table 1: Exponential family components for distributions used in this report

4.2.2 Deriving natural gradients of the ELBO

Let us define the natural parameters for our GMM using mappings η:

$$\gamma_n = \eta_{z_n}(r_n), \quad \nu = \eta_\pi(\alpha), \quad [\lambda_{k1}, \lambda_{k2}]^T = \eta_{\mu_k}\left([m_k, C_k]^T\right), \tag{4.6}$$

noting that $\nu = \alpha$, but we define it uniquely to distinguish ν as a natural parameter. Using the fact that the expectation of the sufficient statistic is the gradient of the log-normaliser $\mathbb{E}[T(x)] = \nabla_\eta A(\eta)$, and Equation 2.9 for the CAVI optimal variational distribution, we can simultaneously rewrite the ELBO as functions of our three natural parameters, while linking this formation to our CAVI updates. Beginning with α, ν:

$$\begin{aligned}
\mathcal{L}(\nu) &= \mathbb{E}_{Z,\pi,\mu}[\ln p(X, Z, \pi, \mu) - \ln q(Z, \pi, \mu)] \\
&= \mathbb{E}_\pi[\ln q^*(\pi|\nu)] - \mathbb{E}_\pi[\ln q(\pi|\nu)] + c \\
&= \nu^*\mathbb{E}[T(\pi)] - A(\nu^*) - \nu\mathbb{E}[T(\pi)] + A(\nu) + c \\
&= \nabla_\nu A(\nu)(\nu^* - \nu) + A(\nu) + c,
\end{aligned} \tag{4.7}$$

where $\nu = [\nu_1, ..., \nu_K]^T$ and $\nu_k^* = N_k + \alpha_0$, the parameterisation of $q^*(\pi)$ derived in Section 2.3 transformed by the natural parameter mapping η. We use the same method to rewrite ELBO as functions of λ_k and γ_n:

$$\mathcal{L}(\lambda_k) = \nabla_{\lambda_k} A(\lambda_k)(\lambda_k^* - \lambda_k) + A(\lambda_k) + c, \tag{4.8}$$

$$\mathcal{L}(\gamma_n) = \nabla_{\gamma_n} A(\gamma_n)(\gamma_n^* - \gamma_n) + A(\gamma_n) + c, \tag{4.9}$$

where $\gamma_n^* = [\gamma_{n1}^*, ...\gamma_{nK}^*]^T = [\ln r_{n1}^*, ... \ln r_{nK}^*]^T$, given by Equations 3.12 and 3.23, and $\lambda_k = [\lambda_{k1}, \lambda_{k2}]^T = \left[C_0^{-1} m_0 + N_k \Sigma^{-1} \bar{x}_k, -\frac{1}{2}\left(N_k \Sigma^{-1} + C_0^{-1}\right)\right]^T$. We can see that these formulations of the ELBO are similar, suggesting a general result, which is that $\nabla_\eta \mathcal{L} = \nabla_\eta A(\eta)(\eta^* - \eta) + A(\eta) + c$, and from which we can differentiate to obtain another general result:

$$\nabla_\eta \mathcal{L} = \nabla_\eta^2 A(\eta)(\eta^* - \eta). \tag{4.10}$$

This states that for each variational natural parameter, the corresponding gradient of the ELBO is the difference between the current and CAVI-determined parameters, pre-multiplied by the Hessian of the log-normaliser.

Let us return to the Fisher information matrix. Using the exponential family representation of q_η, parameterised by $\eta(\phi)$:

$$\begin{aligned} G(\eta) &= \mathbb{E}_\eta[(\nabla_\eta \ln q_\eta)(\nabla_\eta \ln q_\eta)^T] \\ &= \mathbb{E}_\eta\left[(T(\cdot) - \mathbb{E}[T(\cdot)])(T(\cdot) - \mathbb{E}[T(\cdot)])^T\right] \\ &= \mathrm{Cov}(T(\cdot), T(\cdot)) \\ &= \nabla_\eta^2 A(\eta), \end{aligned} \tag{4.11}$$

where we have used the fact that the variance of the sufficient statistic is equal to the Hessian of the log-normaliser. We defined natural gradient earlier as Euclidean gradient premultiplied by the inverse of the Fisher information matrix, therefore the Hessian of the log-normaliser is canceled out, and the natural gradient of the ELBO with respect to any variational natural parameter η is

$$\tilde{\nabla}_\eta \mathcal{L} = \eta^* - \eta. \tag{4.12}$$

Looking back to our CAVI updates, we obtain a truly significant result, namely that setting $\eta = \eta^*$ is equivalent to setting $\eta \leftarrow \eta + \tilde{\nabla}_\eta \mathcal{L}$, which is simply a natural gradient update of fixed step size 1, or a Euclidean gradient update step with step size ρ_t determined automatically by the Fisher information in each iteration.

4.2.3 Stochastic natural gradient descent

By recasting CAVI as natural gradient descent, we can now utilise the mini-batch sampling introduced earlier to reduce computation for a large dataset, and perform stochastic natural gradient descent optimisation with noisy Monte Carlo estimates of the natural gradient:

$$\langle \tilde{\nabla}_{\nu_k} \mathcal{L} \rangle = \frac{1}{S} \sum_{n=1}^{S} (N \hat{r}_{nk} + \alpha_0 - \alpha_k), \tag{4.13}$$

$$\langle \tilde{\nabla}_{\lambda_{k1}} \mathcal{L} \rangle = \frac{1}{S} \sum_{n=1}^{S} \left(C_0^{-1} m_0 + N \hat{r}_{nk} \Sigma^{-1} \hat{x}_n - C_k^{-1} m_k \right), \tag{4.14}$$

$$\langle \tilde{\nabla}_{\lambda_{k2}} \mathcal{L} \rangle = \frac{1}{S} \sum_{n=1}^{S} -\frac{1}{2} \left(N \hat{r}_{nk} \Sigma^{-1} + C_0^{-1} - C_k^{-1} \right). \tag{4.15}$$

Note that the local responsibility updates remain the same as in CAVI, as we have just shown them to be equivalent in the batch case, and they are independent of each other – therefore when mini-batching we simply update a subset of the responsibilities $\{\hat{r}_n\}_{n=1}^S$ in the usual manner; see Algorithm 4.

5 Gradient Estimators

In VI and many other problem domains, the ability to estimate accurate, low variance estimates of function gradients is essential to gradient descent methods. We have so far only considered VI algorithms with distributions within the exponential family, whose gradients (natural or Euclidean) we can calculate analytically, or approximate with 'analytic' gradient estimators that utilise mini-batching. In many real-world applications, this is not feasible, and we must rely on more general gradient estimators with far less information about the distributions being studied.

In VI we are concerned with the gradient of the expectation of the ELBO, and so we focus on

Algorithm 4: Stochastic natural gradient descent VI for a fixed covariance GMM

Input: Data X, number of components K, priors ν_0, λ_0, fixed Gaussian covariance Σ, step size schedule $\rho(t)$
Output: Variational distributions $q(z_n|r_n), q(\pi|\nu), q(\mu_k|\lambda_k)$
Initialise: $t = 0$, variational parameters $r = r_{1:N,1:K}, \nu = \nu_{1:K}, \lambda = \lambda_{1:K}$
while *the* ELBO *has not converged* **do**
  Sample X without replacement, generate subset $X_s \subset X$ of length $S < N$;
  Calculate matrix of responsibilities r following procedure of Algorithm 3;
  for $k \in \{1, ..., K\}$ **do**
    Set $\nu_k \leftarrow \nu_k + \rho(t)\langle\tilde{\nabla}_{\nu_k}\mathcal{L}\rangle$ using Equation 4.13;
    Set $\lambda_{k1} \leftarrow \lambda_{k1} + \rho(t)\langle\tilde{\nabla}_{\lambda_{k1}}\mathcal{L}\rangle$ using Equation 4.14;
    Set $\lambda_{k2} \leftarrow \lambda_{k2} + \rho(t)\langle\tilde{\nabla}_{\lambda_{k2}}\mathcal{L}\rangle$, using Equation 4.15;
  end
  Compute the ELBO(α, m, C, r);
  $t \leftarrow t + 1$;
end

problems of the form

$$\nabla_\theta \mathbb{E}_{p_\theta}[f(x)] = \nabla_\theta \int f(x) p_\theta(x) dx, \tag{5.1}$$

for which we use the Monte Carlo integration to approximate expectations (see Section 7.2.1), and denote $f(x)$ and $p(x)$ the *cost* and the *measure* respectively. Three broad classes of Monte Carlo gradient estimators exist: score function, pathwise, and measure. The latter we do not consider here, see Mohamed et al. 2020 for more information.

5.1 Score function estimator

The score function estimator (SFE), also known as the REINFORCE estimator (R. J. Williams 2004) is an extremely general and widely used gradient estimator that can be loosely considered to be a differentiator of the measure $p_\theta(x)$.

The SFE utilises the score function:

$$\nabla_\theta \ln p(x|\theta) = \frac{\nabla_\theta p(x|\theta)}{p(x|\theta)}, \tag{5.2}$$

which allows us to construct the following MC gradient estimator (Mohamed et al. 2020), provided

we can both sample the measure and evaluate the gradient of its log:

$$\nabla_\theta \mathbb{E}_{p_\theta}[f(x)] = \mathbb{E}_{p_\theta}[f(x)\nabla_\theta \ln p(x|\theta)] \approx \frac{1}{L}\sum_{l=1}^{L} f(\hat{x}^{(l)})\nabla_\theta \ln p(\hat{x}^{(l)}|\theta) = \langle \nabla_\theta \mathbb{E}_{p_\theta}[f(x)]\rangle_{\mathrm{SFE}}, \tag{5.3}$$

where $\hat{x}^{(l)\sim p_\theta(x)}$. In the case of VI we are interested in optimising the ELBO with respect to its variational parameters ϕ:

$$\nabla_\phi \mathcal{L} = \nabla_\phi \int_Z \int_\theta q_\phi(Z,\theta)(\ln p(X,Z,\theta) - \ln q_\phi(Z,\theta))d\theta dZ, \tag{5.4}$$

for which we can construct a SFE, using simple Monte Carlo integration to approximate the entropy term $\mathcal{H}[q_\phi] = \mathbb{E}_{q_\phi}[-\ln q_\phi]$ – we should be able to do this to an abitrary degree of accuracy as long as we can evaluate $\ln q_\phi$.

Unfortunately, score estimators suffer from high variance in their gradient estimates. There are a number of reasons for this (see Mohamed et al. 2020) but intuitively we can see that this estimator uses no information from the cost and knows nothing about its gradient, essentially treating it as a black box. Lacking knowledge about the elements of the cost function means that we must incorporate uncertainty around them into the SFE, even though many of them may not contribute to the gradient. The variance of the gradient estimate is given by $\mathbb{E}[(\nabla_\phi \ln p_\theta(x) f(x))^2] - \langle \nabla_\phi \mathcal{L}\rangle^2$, so we can see that, for example, if $x = \sum_{d=1}^{D} x_d$, the size of the variance will scale with these unseen parameters, $O(D^2)$.

In order to use practically, it is common to apply some form of variance reduction in conjunction with the score function estimator, such as control variates. These replace the black-box cost function of SFE with a substitute cost whose expectation is the same but whose gradient is lower, by introducing a control variate which is correlated with the cost (Mohamed et al. 2020).

The SFE is useful across many areas as a black-box estimator that reduces the need for difficult manual derivations, and has been applied successfully in VI (Paisley et al. 2012; Ranganath et al. 2014), but for our purposes another common type of estimator is more appropriate.

5.2 Pathwise estimator

This estimator is less generalised than the SFE, but when applicable it is able to reduce variance significantly by utilising the structure of the problem. This estimator relies on the reparameterisation trick (Kingma and Welling 2014, which allows us to substitute direct sampling from our measure with samples from a simpler distribution, and transform these samples with a deterministic function, or 'path':

$$\hat{x} \sim p(x|\theta) \quad \equiv \quad \hat{x} = g(\hat{\epsilon}, x), \quad \hat{\epsilon} \sim p(\epsilon). \tag{5.5}$$

With a known base distribution and transformation, we can recast our expectation $\mathbb{E}_{p_\theta}[f(x)] = \mathbb{E}_{p(\epsilon)}[f(g_\theta(\epsilon))]$, and subsequently derive our Monte Carlo estimator:

$$\langle \nabla_\phi \mathcal{L} \rangle_{\mathrm{PW}} = \sum_{l=1}^{L} \nabla_\phi f\left(g\left(\hat{\epsilon}^{(l)}|\theta\right)\right), \quad \hat{\epsilon}^{(l)} \sim p(\epsilon). \tag{5.6}$$

We can see this as effectively moving the dependence of the measure on θ into the cost, allowing the derivative operator to move into the expectation. To apply this estimator we require a path that is differentiable with the chain rule. How do we determine this path? Methods include transformation using the inverse CDF, or for many standard distributions there are 'one-liners' (Devroye 1996) we can apply easily – for instance, for $p_\theta(x) = \mathcal{N}(x|\mu, \Sigma)$ we can simply use $p(\epsilon) = \mathcal{N}(0, I), g(\epsilon) = \mu + \Sigma^{\frac{1}{2}}\epsilon$ (for square root defined by Cholesky decomposition). By directly differentiating the cost function the variance of the gradient estimate can be reduced significantly: only those elements of the cost which affect the gradient are taken into account, and our variance bounds are even independent of dimensionality, unlike the SFE which suffers in high dimensions.

5.3 GMM gradient estimators of the ELBO

We are now in a position to derive score function and pathwise estimators for SGD optimisation of our VI GMM clustering problem. Again, we note that the availability of analytic gradients means you would not use these estimators for this problem in practice, but we wish to compare the effectiveness of a variety of algorithms on a simple, shared problem (see Section 6).

Let us consider our local and global unobserved variables. For these estimators we cannot rely on knowledge about the distributions of the unobserved variables, but we are able to draw

samples from them. As a reminder the ELBO is $\sum_{n=1}^{N} \int q(z_n, \theta|\phi) \left(\ln p(x_n, z_n, \theta) - \ln q(z_n, \theta|\phi)\right) d\theta$. A general purpose Monte Carlo estimate of the ELBO using such samples is below:

$$\langle \mathcal{L} \rangle = \frac{1}{L} \sum_{n,l=1}^{N,L} \ln p(x_n, z_n^{(l)}, \pi^{(l)}, \mu^{(l)}) - \ln q_\phi(z_n^{(l)}, \pi^{(l)}, \mu^{(l)}), \tag{5.7}$$

with samples drawn as follows:

$$z_n^{(l)} \sim q_\phi(z_n) = \text{Mult}(z_n|1, r_n), \tag{5.8}$$

$$\pi^{(l)} \sim q_\phi(\pi) = \text{Dir}(\pi|\alpha), \tag{5.9}$$

$$\mu_k^{(l)} \sim q_\phi(\mu_k) = \mathcal{N}\left(\mu_k|m_k, C_k\right). \tag{5.10}$$

In some VI problems the latent variables are continuous, such as variational autoencoders (VAEs; Kingma and Welling 2014) where they are sampled from a zero-mean Gaussian. In our GMM problem the latent variables are discrete, and their variational parameters r_{nk} are not suitable for gradient descent updates. Therefore for simplicity, and the purposes of comparison, we use Monte Carlo estimates and GD only for π, μ and their variational parameters α, m, C, retaining our analytic 'E step', Equation 3.23, from Section 3.2. This is effectively taking the expectation over Z before estimating the gradient, using $\mathbb{E}[z_{nk}] = r_{nk}$ as usual. While we do not use this in our experiments, we offer a suggested reparameterisation of discrete latent variables in Section 5.3.3.

5.3.1 Score function estimator

Using the sampling distributions of Equations 5.9, 5.10:

$$\langle \nabla_\phi \mathcal{L} \rangle_{\text{SFE}} = \sum_{l=1}^{L} \ln p(X, Z, \pi^{(l)}, \mu^{(l)}) \nabla_\phi \ln q_\phi(Z, \pi^{(l)}, \mu^{(l)}) + \nabla_\phi \mathcal{H}[q_\phi(Z, \pi, \mu)]. \tag{5.11}$$

Recall that we separate the entropy to ensure that the cost function has no dependence on ϕ. In many applications of VI the entropy of the variational gradient can be calculated analytically (Kingma and Welling 2014); for simplicity we assume this here, though we note that finding and using MC gradient estimates of the entropy are not too challenging. The gradient terms for each

variational parameter are:

$$\begin{aligned}
\nabla_{\alpha_k} \ln q(\pi) &= \nabla_{\alpha_k} \left(\ln C(\alpha) + \sum_{k=1}^{K} (\alpha_k - 1) \ln \pi_k \right) \\
&= \psi \left(\sum_{k=1}^{K} \alpha_k \right) - \psi(\alpha_k) + \alpha_k \ln \pi_k, \qquad (5.12) \\
\nabla_{m_k} \ln q(\mu_k) &= -\frac{1}{2} \nabla_{m_k} \left((\mu_k - m_k)^T C_k^{-1} (\mu_k - m_k) + D \ln(2\pi) + \ln |C_k| \right) \\
&= C_k^{-1} (m_k - \mu_k), \qquad (5.13) \\
\nabla_{C_k} \ln q(\mu_k) &= -\frac{1}{2} \left((\mu_k - m_k)(\mu_k - m_k)^T + C_k^{-1} \right). \qquad (5.14)
\end{aligned}$$

5.3.2 Pathwise estimator

We now consider a pathwise estimator approach to our variational GMM. Working from the autoencoding variational Bayes (AEVB) algorithm of Kingma and Welling 2014, we see that we must consider sampling paths for our reparameterisations of variables, $\pi = g(\hat{u}), \mu_k = h(\hat{\xi}_k)$. The AEVB estimator, adapted for the variational GMM, is

$$\langle \nabla_\phi \mathcal{L} \rangle_{\mathrm{PW}} = \frac{1}{L} \sum_{l=1}^{L} \nabla_\phi \left(f(X, Z, g(\hat{\zeta}^{(l)}), h(\hat{\xi}^{(l)}) \right), \qquad (5.15)$$

for cost $f(X, Z, \pi, \mu) = \ln p(X, Z, \pi, \mu) - \ln q_\phi(Z, \pi, \mu)$. To obtain the pathwise estimate we require base sampling distributions and differentiable sampling paths for a Gaussian distribution and a Dirichlet distribution. Starting with μ_k, we derive the sampling paths for differentiatting the cost with respect to $m_k, C_k^{1/2}$ below, beginning with samples from the base distribution $\hat{\xi}_k \sim \mathcal{N}(0, I)$. Note that we use $C_k^{1/2}$ rather than C_k, as it simplifies the differentiation, and ensures positive

definiteness of the covariance matrix is preserved after a gradient descent step.

$$\nabla_{m_k} f = \nabla_{\mu_k} f \nabla_{m_k} \mu_k, \tag{5.16}$$

$$\nabla_{C_k^{1/2}} f = \nabla_{\mu_k} f (\nabla_{C_k^{1/2}} \mu_k)^T, \tag{5.17}$$

$$\begin{aligned} \nabla_{\mu_k} f &= \nabla_{\mu_k} \left(\ln p(X|Z, \mu_k) + \ln p(\mu_k) - \ln q(\mu_k) \right) \\ &= -\sum_{n=1}^{N} \left(r_{nk} \Sigma^{-1} (x_n - \mu_k) \right) - C_0^{-1} (\mu_k - m_0) + C_k^{-1} (\mu_k - m_k), \end{aligned} \tag{5.18}$$

$$\nabla_{m_k} \mu_k = \nabla_{m_k} h(\hat{\xi}_k) = \nabla_{m_k} (m_k + C_k^{1/2} \hat{\xi}_k) = 1, \tag{5.19}$$

$$\nabla_{C_k^{1/2}} \mu_k = \nabla_{C_k^{1/2}} h(\hat{\xi}_k) = \nabla_{C_k} (m_k + C_k^{1/2} \hat{\xi}_k) = \hat{\xi}_k. \tag{5.20}$$

For the Dirichlet reparameterisation a number of methods have been developed, such as approximations with a Weibull distribution (Zhang et al. 2020) and a Gaussian softmax (Srivastava and Sutton 2017), but such approximations can suffer in accuracy. We use the path suggested in Figurnov et al. 2018: firstly samples $\hat{u}_k$ are drawn from a uniform distribution and transformed into Gamma distribution samples $\hat{y}_k$ using the inverse Gamma CDF (shape parameter α_k, rate parameter 1):

$$\hat{y}_k = F_{\alpha_k,1}^{-1}(\hat{u}_k), \hat{u}_k \sim \mathcal{U}(0,1). \tag{5.21}$$

These can then be transformed into samples from the Dirichlet distribution:

$$\hat{\pi}_k = \frac{\hat{y}_k}{\sum_{k=1}^{K} \hat{y}_k}. \tag{5.22}$$

Now we can assemble our derivative with respect to the cost using the chain rule:

$$\nabla_{\alpha_k} f = \nabla_{\pi_k} f \nabla_{\alpha_k} \pi_k = \nabla_{\pi_k} f \nabla_{\hat{y}_k} \pi_k \nabla_{\alpha_k} F_{\alpha_k,1}^{-1}, \tag{5.23}$$

$$\nabla_{\pi_k} f = \nabla_{\pi_k} \left(N_k \ln \pi_k + \ln \pi_k + (\alpha_k - 1) \ln \pi_k \right) = (\alpha_k + N_k)/\pi_k, \tag{5.24}$$

$$\nabla_{\hat{y}_k} \hat{\pi}_k = \frac{\sum_{j=1}^{K} \hat{y}_{j \neq k}}{(\sum_{k=1}^{K} \hat{y}_k)^2}, \tag{5.25}$$

$$\nabla_{\alpha_k} F_{\alpha_k,1}^{-1} = \frac{1}{\epsilon} \left(F_{\alpha_k+\epsilon,1}^{-1}(\hat{u}_k) - F_{\alpha_k,1}^{-1}(\hat{u}_k) \right). \tag{5.26}$$

Following the work of Joo et al. 2019; Knowles 2015, we use a high accuracy numerical approx-

imation of the inverse CDF using SciPy's `stats.gamma.ppf` function (Virtanen et al. 2020), and use the finite difference method to calculate its gradient.

5.3.3 Reparameterising discrete latent variables

As an aside, we note that it is possible to reparameterise the one-hot samples of z_n from the categorical distribution using a suitable approximation. Jang et al. 2017 make use of the Gumbel trick to approximate discrete distributions, replacing the undifferentiable argmax with a softmax to define the differentiable Gumbel-softmax distribution. The path for r_{nk} is given below:

$$\nabla_{r_{nk}} f = \nabla_{z_{nk}} f \nabla_{r_{nk}} z_{nk}, \tag{5.27}$$

$$\nabla_{r_{nk}} z_{nk} = \nabla_{r_n k} \left(\frac{\exp((\ln r_{nk} + \hat{\epsilon}_k)/\tau)}{\sum_{k=1}^{K} \exp((\ln r_{nk} + \hat{\epsilon}_k)/\tau)} \right), \tag{5.28}$$

$$\begin{aligned} \nabla_{z_{nk}} f &= \nabla_{z_{nk}} \left(-\frac{1}{2} z_{nk} (x_n - \mu_k)^T \Sigma^{-1} (x_n - \mu_k) + z_{nk} \ln \pi_k + z_{nk} \ln r_{nk} \right) \\ &= -\frac{1}{2} (x_n - \mu_k)^T \Sigma^{-1} (x_n - \mu_k) + \ln \pi_k + \ln r_{nk}, \end{aligned} \tag{5.29}$$

where ϵ_k is a sample of the Gumbel distribution using the Gumbel trick $\hat{\epsilon}_k = -\ln(-\ln(\hat{u}_k)), \hat{u}_k \sim \mathcal{U}(0,1)$ and τ is a temperature parameter. As $\tau \to 0$, draws from the Gumbel-softmax become one-hot vectors, but there is a trade-off in gradient variance. This is an interesting area for future research, and the author's results rival traditional VAEs, but we do not consider it further here.

5.4 Alternative lower bounds

Although we focus on the ELBO in this report, there is a lot of research into alternative variational lower bounds that reduce gradient variance, particularly in the case of VAEs. The best-known and simplest of these is the importance weighted autoencoder (IWAE; Burda et al. 2016), which can be used in many non-VAE VI applications, including the variational GMM. The idea is simple –

beginning with a Monte Carlo estimator of the ELBO:

$$\langle \mathcal{L}(\theta, \phi) \rangle_{\text{VAE}} = \frac{1}{L} \sum_{l=1}^{L} \ln \frac{p\left(x, \theta^{(l)}\right)}{q_\phi(\theta^{(l)})} \leq \ln p(x), \tag{5.30}$$

$$\langle \mathcal{L}(\theta, \phi) \rangle_{\text{IWAE}} = \ln \frac{1}{K} \sum_{k=1}^{K} \frac{p\left(x, \theta^{(l)}\right)}{q_\phi(\theta^{(l)})} \leq \ln p(x), \tag{5.31}$$

we swap the order of the log and the weighted summation, and by Jensen's inequality can say that $\langle \mathcal{L} \rangle_{\text{VAE}} \leq \mathcal{L}_{\text{IWAE}}(\theta, \phi)$. IWAE is therefore a tighter, and therefore superior, lower bound on the evidence. Below we derive the gradient estimator using IWAE, using two applications of the score function on lines two and three:

$$\begin{aligned}
\langle \nabla_\phi \mathcal{L} \rangle_{\text{IWAE}} &= \nabla_\phi \ln \frac{1}{L} \sum_{l=1}^{L} \frac{p\left(x, \theta^{(l)}\right)}{q_\phi(\theta^{(l)})} \\
&= \frac{\sum_{l=1}^{L} \nabla_\phi p\left(x, \theta^{(l)}\right) / q_\phi(\theta^{(l)})}{\sum_{l=1}^{L} p(x, \theta^{(l)}) / q_\phi(\theta^{(l)})} \\
&= \sum_{l=1}^{L} \frac{p\left(x, \theta^{(l)}\right) / q_\phi(\theta^{(l)})}{\sum_{l=1}^{L} p(x, \theta^{(l)}) / q_\phi(\theta^{(l)})} \nabla_\phi \ln \frac{p\left(x, \theta^{(l)}\right)}{q_\phi(\theta^{(l)})} \\
&= \sum_{l=1}^{L} w_l \nabla_\phi \ln \frac{p\left(x, \theta^{(l)}\right)}{q_\phi(\theta^{(l)})}.
\end{aligned} \tag{5.32}$$

We can understand this tighter bound intuitively by comparing with the Monte Carlo gradient estimate of the ELBO. For IWAE, rather than weighting each sample equally by number of samples L, they are more appropriately weighted by their relative importance w_l. Further work improves on IWAE (Roeder et al. 2017), most notably doubly reparameterised gradient estimators (DREG; Tucker et al. 2018), which addresses a hidden, high variance score-function term in the IWAE pathwise gradient by performing another reparameterisation on it. Due to time and space constraints, we do not consider these estimators further in this report, or in our experiments.

6 Experiments

In this section we compare some of the approaches to variational inference detailed in this report, using the GMM clustering problem detailed in Section 3. We are primarily interested in comparing

convergence rates, in both wall-clock time and number of iterations, between methods and with varying sample sizes.

These experiments were all implemented in Python using NumPy and SciPy. No automatic machine learning libraries such as Scikit-learn were used (Pedregosa et al. 2011), and all VI algorithms were written from scratch. All code used in this project can be found at https://github.com/BenGutteridge/4YP, as well as gif animations of the experimental runs in this section.

While the ELBO is the value of interest for the majority of this report, it is of little use as a convergence diagnostic or comparitive performance metric. As noted by Yao et al. 2018, comparing the ELBO between different methods is meaningless, and the value itself is unintuitive. The authors propose their own diagnostics, but these are unnecessarily complicated for our investigation. An alternative might be the (symmetrised) KL divergence, but this is reflected in the ELBO. We choose log-likelihood error as a simple and intuitive measure of error: $\ln p(X|\pi_{\text{true}}, \mu_{\text{true}}) - \ln p(X|\mathbb{E}[\pi], \mathbb{E}[\mu])$, comparing the ground truth GMM parameters with our best approximations.

Our synthetic dataset consists of $N = 700$ points with 7 clusters; we start with $K = 10$ mixture components and allow the algorithms to eliminate unnecessary components automatically by driving mixture weights to zero.

6.1 Comparing VI approaches

The five key VI algorithms that were built and that are explored here are coordinate ascent variational inference (CAVI; equivalent to natural gradient descent), gradient descent (GD), stochastic gradient descent (SGD), stochastic natural gradient descent (SNGD) and stochastic gradient descent using the pathwise estimator (PW). An algorithm for SGD using the SFE was also written, but as mentioned in Section 5, this method was found to be unsuitable, failing to converge due to the N^2 scaling of gradient estimate variance; hence we leave out the SFE here.

Figure 6.1 shows an illustrative set of runs over our five methods, both in wall-clock time and iterations. As expected the pathwise estimator performs worst: it is a general method and the only one that does not use an analytic evaluation of the expectation of the ELBO gradient; it must evaluate over multiple samples to perform Monte Carlo integration, and uses the entire dataset.

CAVI also evaluates over the entire dataset, but does not require multiple samples, so it is faster in wall-clock time as well as being by far the most accurate approach and the fastest in terms of iterations. Both SNGD and SGD are fast and accurate, though SGD requires more iterations.

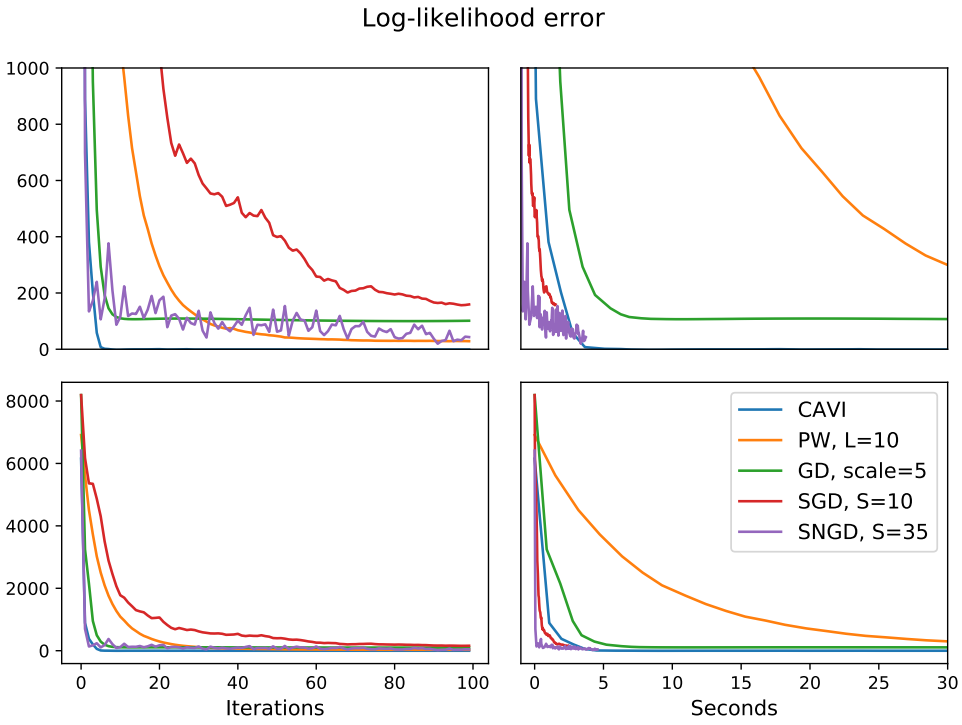

Figure 6.1: Log-likelihood error of all multiple VI approaches with illustrative runs. L, S are number of samples and size of mini-batch, respectively

6.2 Step size schedules for GD

For simplicity our SGD and PW approaches use an identical GD step schedule, with heuristically determined base step sizes for each variational parameter that decay exponentially: $\rho_t = s \cdot \exp(-0.01t)$, where s is a scale parameter that we set to 1. The benefit of accurate gradients is that we can use larger step sizes in GD schemes; we look at this in Figure 6.2 by varying s. We find that the speed and accuracy increase for larger steps, though there is a risk of instability, as seen in the case of $s = 10$, where performance is initially strong but begins to wander. The $s = 20$ case did not even initially converge, so it is not plotted.

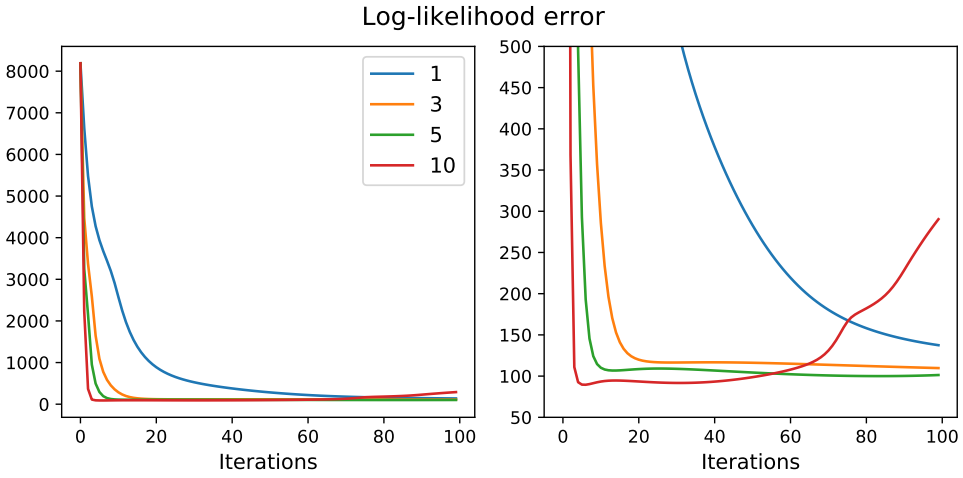

Figure 6.2: Log-likelihood error of GD runs with various numbers of scales s of GD schedule $\rho_t = s \cdot \exp(-0.01t)$

6.3 Mini-batch sizes

In Figure 6.3 we compare the convergence rates of various mini-batch sizes for SGD. We can see that while the large mini-batches do converge in fewer iterations, and ultimately have lower error, smaller mini-batches can reach a similar level of accuracy in much less time. Unfortunately, it is inconvenient to determine base step sizes and ideal mini-batch sizes in advance.

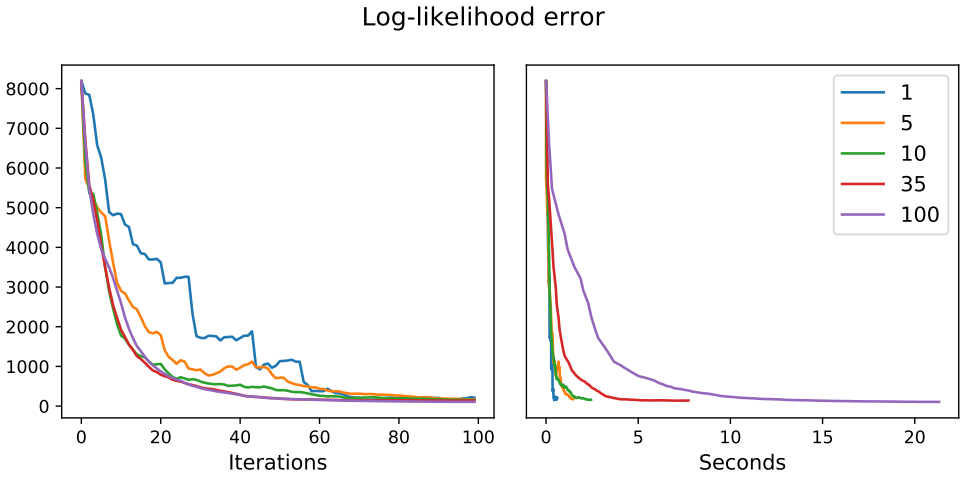

Figure 6.3: Log-likelihood error of SGD runs with various sized mini-batches

Figure 6.4 shows similar results for SNGD, which are furstratingly poor compared with SGD, likely due to the untuned step size. We show in Section 4 that CAVI, our most efficient VI algorithm, is equivalent to natural gradient descent for step size 1, regardless of parameter. Thus we use

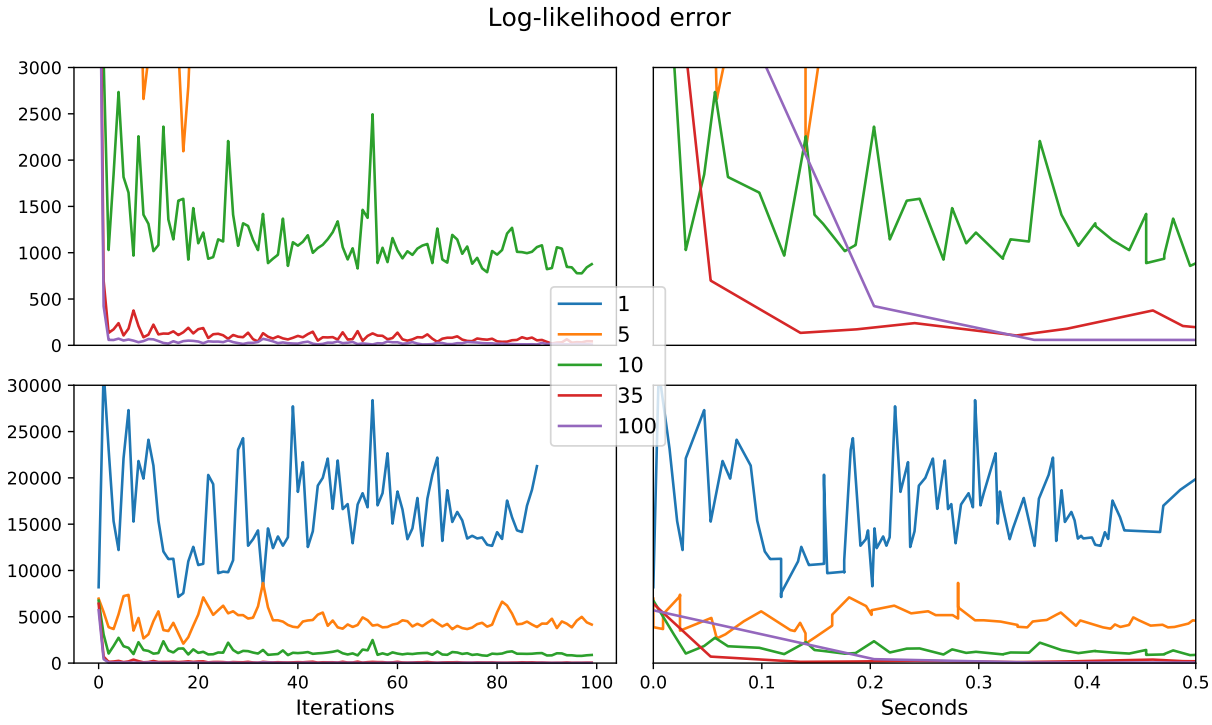

Figure 6.4: Log-likelihood error of SNGD runs with various sized mini-batches

a base step size of 1 for all ϕ and decay exponentially, though this results in some large jumps, particularly for smaller mini-batches.

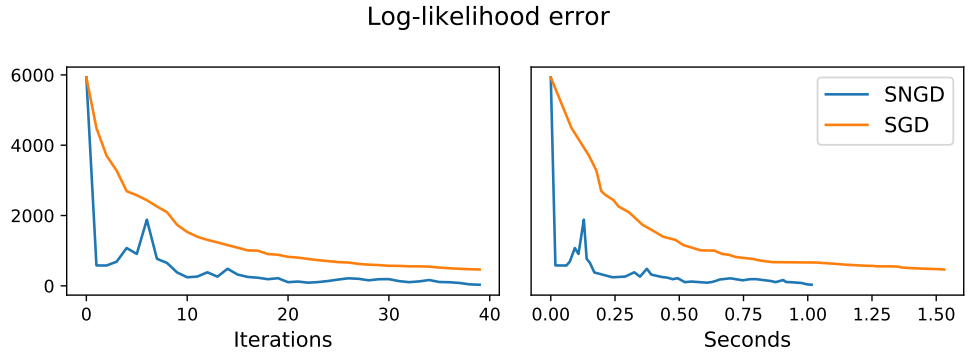

Figure 6.5: Comparing a single run each of SNGD and SGD, each using a mini-batch of 10 samples per iteration

The main benefits of natural over traditional gradients are demonstrated in Figure 6.5. We can see that the SNGD converges more accurately in fewer iterations, as we would expect since the natural gradient is a more appropriate optimisation path for parameters of probability distributions (see Section 4.2). Furthermore, each SNGD iteration is faster than an SGD iteration, as in the latter

we effectively must compute the Fisher information matrix each time.

6.4 Number of Monte Carlo samples

In Figure 6.6 we compare PW runs over multiple sample sizes, and show that single sample is generally sufficient for convergence, and optimal in terms of wall-clock time. When utilising the IWAE estimator, results shown in Figure 6.7, results are similar, the single sample covering more quickly as expected due to requiring fewer evaluations, but we see slightly faster initial convergence using multiple samples. This is not as fast as we would like, but the distributions sampled in this problem are quite simple – for more complex distributions, importance weighting will be more impactful.

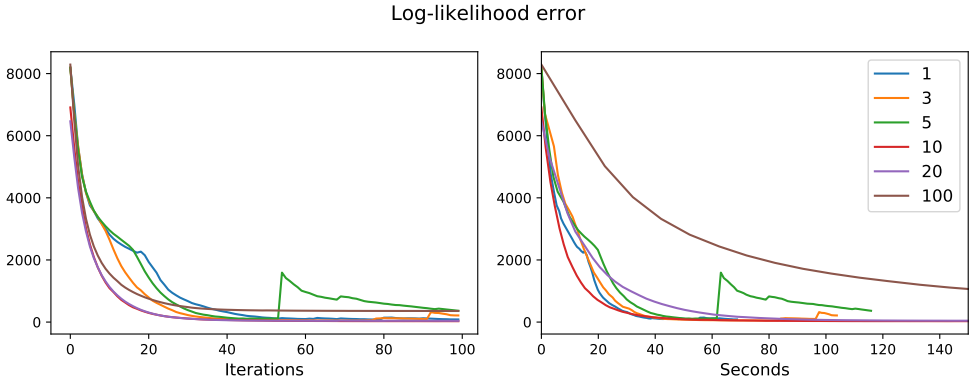

Figure 6.6: Log-likelihood error of PW runs with various numbers of samples used per iteration

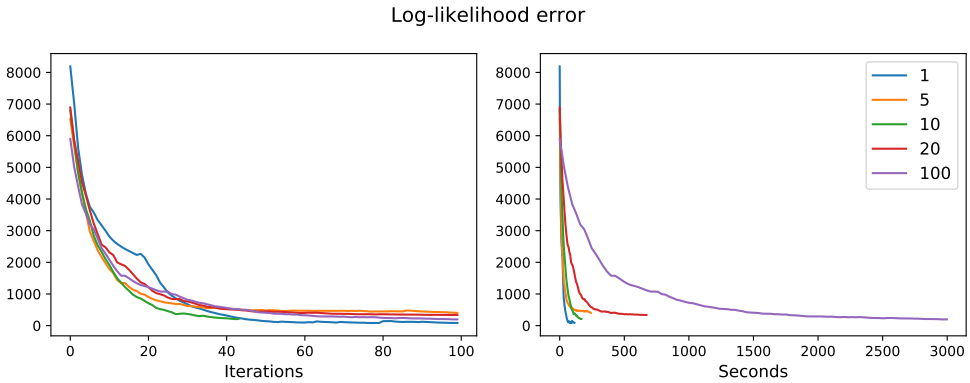

Figure 6.7: Log-likelihood error of PW runs with IWAE and various numbers of samples

7 Gaussian Processes and Bayesian Quadrature

In this section, we explore Gaussian processes (GPs) and their applications – GP regression, Bayesian quadrature, and Bayesian optimisation – to help address the generic problem of variational inference. Unfortunately, time constraints limited the amount of experimentation that was possible in this area, and some avenues of interest were found to be unworkable after investigation. We include these 'dead ends' to demonstrate the work put into investigation, literature review and understanding of these topics, and pose suggested avenues for future research.

7.1 Gaussian processes

Intuitively, a GP is a distribution over functions. If we think of an unknown function $f(x)$ as an infinite-length vector of outputs for infinite inputs $x \in \mathbb{R}$, then there is no reason we cannot imagine $f(x)$ as a 'random function' distributed by an infinite-dimensional distribution. A Gaussian process can be thought of as an infinite-dimensional extension of the multivariate Gaussian distribution; a collection of random variables over a continuous domain. The beauty of GPs is that any finite number of these random variables are jointly Gaussian distributed. For example, a function evaluation at a single point is distributed by a Gaussian: $f(x_1) \sim \mathcal{N}(\mu, \sigma)$, a susbet of two evaluations are distributed by a 2D Gaussian $f(x_1, x_2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and so on. The 'marginalisation property' means that distributional parameters between larger and smaller sets of variables are consistent: $\Sigma_{11} = \sigma$, for instance (C. E. Rasmussen and C. K. I. Williams 2006).

A GP is defined entirely by its mean and covariance functions:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')), \tag{7.1}$$

$$m(x) = \mathbb{E}[f(x)], \tag{7.2}$$

$$k(x, x') = \mathrm{Cov}(f(x), f(x')) = \mathbb{E}[(f(x) - m(x))(f(x') - m(x')]. \tag{7.3}$$

We often choose $m(x) = 0$ and allow the GP to be defined entirely by its covariance, or kernel, function. A typical choice is the squared exponential or radial basis function kernel:

$$k_{SE}(x, x') = \sigma_s^2 \exp\left(\frac{-(x - x')^2}{2\ell^2}\right). \tag{7.4}$$

This essentially acts as a smoothing function, ensuring neighbouring points are closely correlated, and is defined by two hyperparameters: the lengthscale ℓ which essentially defines the smoothness, or the breadth of the region of correlation between neighbouring points, and the output variance σ_s, a scale factor common to all kernel functions.

7.1.1 Gaussian process regression

The flexibility of GPs and their ability to represent uncertainty makes them ideal function approximators. By beginning with a GP prior and performing posterior inference over a number of function evaluations, we can approximate the unknown (but evaluatable) function. This is extremely useful in situations where evaluating the unknown function is expensive. We give a brief overview of training a Gaussian process on function evaluations below.

We can draw a sample from a GP by denoting a set of prediction points and sampling from the GP at those points. We denote a function draw from the GP prior using prediction points X_* with f_*:

$$f_* \sim \mathcal{N}(0, K(X_*, X_*)). \tag{7.5}$$

Here we have assumed a zero mean function, as in Figure 7.1. Before training, these function draws are uninformative. Now let us consider the function draws conditioned on a set of true function evaluations X, f: this is our training data. For noise free evaluations, we can express our posterior GP as (C. E. Rasmussen and C. K. I. Williams 2006):

$$f_* \mid X_*, X, f \sim \mathcal{N}\left(\bar{f}_*, \mathrm{Cov}(f_*)\right), \tag{7.6}$$

$$\bar{f}_* = K\left(X_*, X\right) K(X, X)^{-1} f, \tag{7.7}$$

$$\mathrm{Cov}(f_*) = K\left(X_*, X_*\right) - K\left(X_*, X\right) K(X, X)^{-1} K\left(X, X_*\right), \tag{7.8}$$

where we have defined the posterior mean and covariance functions as $\bar{f}_*$ and $\mathrm{Cov}(f_*)$ respectively. As Figure 7.1 demonstrates, GPs are a powerful regression tool, providing not only a reasonable interpolation of the unknown function in untested regions, but also uncertainty information in those regions.

GP regression serves as a springboard into a variety of other uses of GPs, such as in classi-

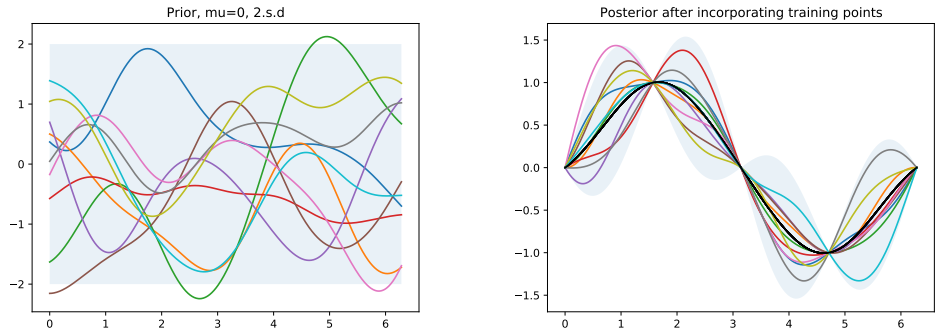

Figure 7.1: Prior and posterior for 500 predicted points X_* and 5 equally spaced training points X evaluated from a sine function, with 10 drawn function samples shown for each, and error bars showing two standard deviations from the posterior mean function. A zero mean function and SE kernel with $\ell = 1, \sigma_s = 1$ has been used.

fication, deep networks, as priors in spatiotemporal data modelling, and in Bayesian optimisation (Damianou and Lawrence 2013; Frazier 2018; Simpson et al. 2017; Van der Wilk et al. 2017). In the next section, we look at Bayesian quadrature (BQ), a probabilistic method of approximating intractable integrals using GPs.

7.2 Numerical integration

As discussed previously, the core problem of posterior inference is often the computation of an intractable integral:

$$\int f(x)p(x)dx = \mathbb{E}_{p(x)}[f(x)]. \tag{7.9}$$

For instance, the computation of a model evidence $p(D) = \int p(\theta|D)p(\theta)d\theta$ as the normalising constant in a posterior, or a posterior predictive $p(x^*|D) = \int p(x^*|D,\theta)p(\theta|D)d\theta$. Variational inference is one method of addressing this problem, and is advantageous in that it produces both an estimate of the posterior and the evidence simultaneously in the form of the variational distribution q and the ELBO respectively. The other methods can be divided into Monte Carlo and Bayesian quadrature.

7.2.1 Monte Carlo integration

Recall Simple Monte Carlo (SMC) integration (using $\mathcal{Z}$ to denote the unknown integral):

$$\mathcal{Z} = \int f(x)p(x)dx \approx \frac{1}{N}\sum_{l=1}^{L} f(\hat{x}^{(l)}), \quad \hat{x}^{(l)} \sim p(x). \tag{7.10}$$

This is an unbiased estimator of the true integral value, and using N samples $\hat{x}$ drawn from the known distribution $p(x)$, SMC will converge to the integral as $N \to \infty$ by the strong law of large numbers (Robert and Casella 2004). Typically in real applications, $p(x)$ will be difficult to sample, or samples drawn from it will correlate poorly with informative regions of $f(x)$ (e.g. likelihoods with narrow peaks). Variants on SMC are used in these cases, such as importance sampling or Markov-chain Monte Carlo methods (Hastings 1970), but they have their own issues. For instance, importance sampling relies on a sampling distribution $\tilde{q}(x)$ and weights samples by their relevance,

$$\int f(x)\frac{p(x)}{\tilde{q}(x)}\tilde{q}(x)dx, \tag{7.11}$$

then performs SMC using samples from $\tilde{q}(x)$ with an augmented $\tilde{f}(x) = f(x)p(x)/\tilde{q}(x)$. As explained in O'Hagan 1987, this is a common but statistically unsound way to approximate integrals, as the sampling distribution is chosen arbitrarily, and value of the approximation would vary for differing $\tilde{q}_1, \tilde{q}_2$, even if the same set of samples were drawn from them. Furthermore, the actual sampled values $\hat{x}$ are ignored: if the same $\hat{x}$ were sampled twice, each sample would be treated as equally informative, despite providing no new information about the unknown integrand. BQ (Ghahramani and C. Rasmussen 2002; O'Hagan 1991) addresses these problems with a Bayesian treatment of the unknown expectation $\mathcal{Z}$.

7.2.2 Bayesian quadrature

Whereas SMC can be thought of as integration using an approximation of the prior (using a set of samples from $p(x)$ rather than a true integration over it), BQ can be imagined integration using an approximation of $f(x)$. A Gaussian process is trained on a set of evaluations f_s to approximate f,

and then we can analytically evaluate the following integral using standard Gaussian identities:

$$\mathbb{E}[\mathcal{Z}|\boldsymbol{f}_s] = \int \bar{f}_* p(x) dx, \tag{7.12}$$

where $p(x)$ is typically a Gaussian or mixture of Gaussians; alternatively an importance re-weighting can be used for difficult to sample $p(x)$, similar to Equation 7.11, provided $p(x)$ can be evaluated. Improvements on classical BQ include active sampling techniques, marginalising over GP hyper-parameters, and warping the GP with a log or square root to better represent the case when $f(x)$ is a likelihood (Gunter et al. 2014; Osborne et al. 2012). We consider active sampling in Section 7.3.4.

7.3 BQ in variational inference

We turn our attention to the uses of BQ in variational inference, looking at Variational Bayesian Monte Carlo (VBMC) and considering further research areas. We also consider BQ as a method of performing intelligent subsampling for mini-batching large datasets.

7.3.1 Variational Bayesian Monte Carlo

As a reminder, our general expression for the ELBO is:

$$\begin{aligned} \mathcal{L} &= \int q_\phi(\theta) \ln \left(\frac{p(D,\theta)}{q_\phi(\theta)} \right) d\theta \\ &= \mathbb{E}_\theta[\ln p(D,\theta)] + \mathcal{H}[q_\phi]. \end{aligned} \tag{7.13}$$

The ELBO is an integral, and often an intractable one, so we can use BQ to probabilistically estimate it. Using BQ specifically for the purpose of VI was introduced by Acerbi 2018; Acerbi 2020 in their VBMC algorithm. Their motivation is in using VI to approximate the posterior $p(\theta|D) = \frac{p(\theta,D)}{p(D)}$ with an expensive-but-possible to evaluate joint distribution (or unnormalised posterior) and intractable model evidence $p(D) = \int p(D|\theta)p(\theta)d\theta$. Samples of model parameters θ_s are chosen and evaluated to generate training data $\theta_s, f_s = f(\theta)_s$, which is used to train a GP using the method described in the previous section. Using standard Gaussian identities, the expectation over the GP-approximated log-joint and the gradient of this expectation can be evaluated analytically The

variational entropy term and its gradient can also be cheaply and accurately approximated using MCMC, therefore the ELBO can be approximated in a far more Bayesian manner than standard Monte Carlo integration. Finally the gradients are used to perform SGD optimisation on the ELBO as usual.

7.3.2 BQ for gradient estimation

VBMC is a highly developed VI framework, but it is the only one of its kind to utilise BQ. Further research into the applications of GPs in VI is necessary – one potential area of interest might be in using BQ to estimate the gradient directly, as opposed to just the ELBO, to obtain low variance estimates. This might involve utilisation of lower variance estimators like IWAE and DReG (Burda et al. 2016; Tucker et al. 2018, see Section 5). Another area of interest is in utilising correlations between successive gradient updates. In each variational update the log-joint is evaluated, but it stays the same – only the variational distribution changes. Could information from previous updates be used to improve the current gradient estimate?

7.3.3 Intelligently subsampling data

One area we considered when working on this report was that of subsampling large datasets intelligently; more specifically, how can we improve on the common practice of using random sampling in the calculation of gradient estimates for gradient descent optimisation? Randomness is suboptimal; an intelligent Bayesian approach is likely to produce more accurate estimates with fewer samples, and provide more accurate uncertainty information.

Considering the ELBO for a VI problem:

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \int q_\phi(Z_n, \theta) \ln \left(\frac{p(X_n, Z_n, \theta)}{q_\phi(X_n, \theta)} \right) d\theta. \tag{7.14}$$

Here we use X_n, Z_n to refer to a single sample $x_n \in X, z_n \in Z$ repeated N times, as used for our SGD and SNGD implementations in Section 4. We can recognise the ELBO and its gradient as

simple Monte Carlo approximations over the data distribution $p(X)$:

$$\langle \mathcal{L} \rangle = \frac{1}{S} \sum_{n=1}^{S} \mathcal{L}(X_n) \approx \int \mathcal{L}(X, Z, \theta) p(X) dX, \tag{7.15}$$

$$\langle \nabla_\phi \mathcal{L} \rangle = \frac{1}{S} \sum_{n=1}^{S} \nabla_\phi \mathcal{L}(X_n) \approx \int \nabla_\phi \mathcal{L}(X, Z, \theta) p(X) dX. \tag{7.16}$$

Previously studied method of data subsampling, particularly in the case of gradient estimates, include the Horivitz-Thompson gradient estimator (Clemencon et al. 2014; Clémençon et al. 2015), where each sample x_i is accepted with a probability $\pi_i(x)$ and weighted by that probability, but in practice these weighting probabilities are hard to obtain.

We could easily train a GP on evaluations of a subset $S \subset X$ to approximate the ELBO function or any of its gradients – this would be cheaper than evaluating over the entire dataset. However for BQ to be more effective than Monte Carlo, we need to be able to analytically evaluate $\int \mathcal{L} \cdot p(X) dX$, and $p(X)$ is the exact model evidence we are trying to approximate by maximising the ELBO. Approximating this integral with SMC over S yields the same result as our original integral over $\mathcal{L}$ as the GP is trained to pass through those points – not an interesting result, for the trouble of training a Gaussian process (complexity $\mathcal{O}(S^3)$), but we do gain access to the apparatus of active sampling used in GPs.

7.3.4 Active sampling

BQ is fundamentally a sequential sampling problem. When training the GP, we aim to choose samples that will be most informative about the function we are approximating. Methods for doing this include minimising the expected entropy of the integral, i.e. minimising the expected variance of the integral $\mathcal{Z}$ (Osborne et al. 2012), or uncertainty sampling; this is choosing samples that maximise the uncertainty of the integrand (Gunter et al. 2014):

$$x_* = \arg\min_x \langle \mathrm{Var}_{f|s}[\mathcal{Z}] \rangle, \tag{7.17}$$

$$x_* = \arg\max_x \mathrm{Var}_{f|s}[f(x)p(x)]. \tag{7.18}$$

Unfortunately $\mathrm{Var}_{f|s}[\mathcal{Z}] = \iint \mathrm{Cov}(f_*)p(x)p(x')dxdx'$, so to be solved analytically both of these methods require knowledge of the target evidence $p(X)$ – useless to us for a VI problem.

In Monte Carlo integration we can obtain an unbiased estimate of the variance, from which we can estimate error bars of our integral approximation:

$$\mathrm{Var}(f) = \frac{1}{S-1}\sum_{n=1}^{S}\left(f(x_n) - \mathcal{Z}\right), \tag{7.19}$$

$$\mathrm{Var}(\mathcal{Z}) = \frac{\mathrm{Var}(f)}{S}. \tag{7.20}$$

It may be that we can choose a more representative subsample S by choosing $x_* = \arg\max_x \mathrm{Var}(f_*)$, i.e. by evaluating the diagonal values of the GP covariance matrix over X and choosing x_* which has the greatest variance. This is a form of uncertainty sampling more akin to the acquisition functions of Bayesian optimisation (Agnihotri and Batra 2020): cheap to optimise functions which iteratively choose sampling points for GP training with some balance of exploration and exploitation. This method would be equivalent to an Upper Confidence Bound acquisition function $\alpha(x) = \lambda\bar{f}_*(x) + \mathrm{Var}(f_*(x))$ with $\lambda = 0$, i.e. exploration only.

8 Conclusion

Variational inference is a crucial inference tool as it provides a fast Bayesian alternative to inefficient random sampling methods. As an optimisation problem, most VI uses gradient descent, and by considering VI algorithms with a focus on gradients, we can better compare different approaches. In this report, we have provided an overview of gradient estimation techniques and gradient descent VI algorithms, including natural gradients, and demonstrated the link between the traditional CAVI approach and gradient methods. We have produced VI algorithms from scratch for a GMM point clustering model and demonstrated some of the strengths and weaknesses of various methods, considering accuracy and speed. VI is a thriving area of ongoing research, and future work in this area may consider the use of Bayesian quadrature for lower variance gradient estimation or intelligent subsampling for Gaussian process regression, as well as lower variance and problem-specific variational lower bounds.

References

Acerbi, L. (2018). Variational Bayesian Monte Carlo. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/747c1bcceb6109a4ef936bc70cfe67de-Paper.pdf

Acerbi, L. (2020). Variational Bayesian Monte Carlo with Noisy Likelihoods. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (pp. 8211–8222). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/5d40954183d62a82257835477ccad3d2-Paper.pdf

Agnihotri, A., & Batra, N. (2020). Exploring Bayesian optimization [https://distill.pub/2020/bayesian-optimization]. *Distill*. https://doi.org/10.23915/distill.00026

Amari, S.-i. (1998). Natural Gradient Works Efficiently in Learning. *Neural Computation*, *10*(2), 251–276. https://doi.org/10.1162/089976698300017746

Arnott, R. D. (1980). Cluster analysis and stock price comovement. *Financial Analysts Journal*, *36*(6), 56–62. http://www.jstor.org/stable/4478401

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Burda, Y., Grosse, R., & Salakhutdinov, R. (2016). Importance weighted autoencoders.

Clemencon, S., Bertail, P., & Chautru, E. (2014). Scaling up M-estimation via sampling designs: The Horvitz-Thompson stochastic gradient descent. *2014 IEEE International Conference on Big Data (Big Data)*, 25–30. https://doi.org/10.1109/BigData.2014.7004208

Clémençon, S., Bellet, A., Jelassi, O., & Papa, G. (2015). Scalability of Stochastic Gradient Descent based on "Smart" Sampling Techniques. *Procedia Computer Science*, *53*, 308–315. https://doi.org/10.1016/j.procs.2015.07.308

Damianou, A., & Lawrence, N. D. (2013). Deep Gaussian processes. *Artificial intelligence and statistics*, 207–215.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, *39*(1), 1–22. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x

Devroye, L. (1996). Random variate generation in one line of code. *Winter Simulation Conference*.

Di Marco, A., & Navigli, R. (2013). Clustering and diversifying web search results with graph-based word sense induction. *Computational Linguistics*, *39*(3), 709–754.

Figurnov, M., Mohamed, S., & Mnih, A. (2018). Implicit reparameterization gradients. *arXiv preprint arXiv:1805.08498*.

Filipovych, R., Resnick, S. M., & Davatzikos, C. (2011). Semi-supervised cluster analysis of imaging data. *NeuroImage*, *54*(3), 2185–2197.

Frazier, P. I. (2018). A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Geman, S., & Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 721–741.

Ghahramani, Z., Beal, M. J. et al. (1999). Variational inference for Bayesian mixtures of factor analysers. *NIPS*, *12*, 449–455.

Ghahramani, Z., & Rasmussen, C. (2002). Bayesian Monte Carlo. *Advances in Neural Information Processing Systems*, *15*, 505–512. Retrieved January 30, 2021, from https://proceedings.neurips.cc/paper/2002/hash/24917db15c4e37e421866448c9ab23d8-Abstract.html

Gunter, T., Osborne, M. A., Garnett, R., Hennig, P., & Roberts, S. J. (2014). Sampling for inference in probabilistic models with fast Bayesian quadrature. *arXiv preprint arXiv:1411.0439*.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, *57*(1), 97–109. https://doi.org/10.1093/biomet/57.1.97

Hoffman, M. D., Blei, D. M., Wang, C., & Paisley, J. (2013). Stochastic Variational Inference. *Journal of Machine Learning Research*, *14*(4), 1303–1347. http://jmlr.org/papers/v14/hoffman13a.html

Jang, E., Gu, S., & Poole, B. (2017). Categorical Reparameterization with Gumbel-Softmax [arXiv: 1611.01144]. *arXiv:1611.01144 [cs, stat]*. Retrieved April 10, 2021, from http://arxiv.org/abs/1611.01144

Jebara, T. (2012). *Machine learning: Discriminative and generative* (Vol. 755). Springer Science & Business Media.

Joo, W., Lee, W., Park, S., & Moon, I.-C. (2019). Dirichlet Variational Autoencoder [arXiv: 1901.02739]. *arXiv:1901.02739 [cs, stat]*. Retrieved April 9, 2021, from http://arxiv.org/abs/1901.02739

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. (1998). An Introduction to Variational Methods for Graphical Models. In M. I. Jordan (Ed.), *Learning in Graphical Models* (pp. 105–161). Springer Netherlands. https://doi.org/10.1007/978-94-011-5014-9_5

Karlgren, J. (1994). *Newsgroup clustering based on user behavior-a recommendation algebra*. Swedish Institute of Computer Science.

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes [arXiv: 1312.6114]. *arXiv:1312.6114 [cs, stat]*. Retrieved January 30, 2021, from http://arxiv.org/abs/1312.6114

Knowles, D. A. (2015). Stochastic gradient variational Bayes for gamma approximating distributions [arXiv: 1509.01631]. *arXiv:1509.01631 [stat]*. Retrieved April 10, 2021, from http://arxiv.org/abs/1509.01631

Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86. https://doi.org/10.1214/aoms/1177729694

Mohamed, S., Rosca, M., Figurnov, M., & Mnih, A. (2020). Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, *21*(132), 1–62. Retrieved February 1, 2021, from http://jmlr.org/papers/v21/19-346.html

Neal, R. M., & Hinton, G. E. (1998). A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants. In M. I. Jordan (Ed.), *Learning in Graphical Models* (pp. 355–368). Springer Netherlands. https://doi.org/10.1007/978-94-011-5014-9_12

O'Hagan, A. (1987). Monte Carlo is Fundamentally Unsound [Publisher: [Royal Statistical Society, Wiley]]. *Journal of the Royal Statistical Society. Series D (The Statistician)*, *36*(2/3), 247–249. https://doi.org/10.2307/2348519

O'Hagan, A. (1991). Bayes–Hermite quadrature. *Journal of Statistical Planning and Inference*, *29*(3), 245–260. https://doi.org/10.1016/0378-3758(91)90002-V

Osborne, M., Garnett, R., Ghahramani, Z., Duvenaud, D. K., Roberts, S. J., & Rasmussen, C. (2012). Active Learning of Model Evidence Using Bayesian Quadrature. *Advances in Neural Information Processing Systems*, *25*, 46–54. Retrieved January 29, 2021, from https://proceedings.neurips.cc/paper/2012/hash/6364d3f0f495b6ab9dcf8d3b5c6e0b01-Abstract.html

Paisley, J., Blei, D., & Jordan, M. (2012). Variational Bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Petersen, K., Pedersen, M. et al. (2008). The matrix cookbook, vol. 7. *Technical University of Denmark*, *15*.

Ranganath, R., Gerrish, S., & Blei, D. (2014). Black Box Variational Inference. In S. Kaski & J. Corander (Eds.), *Proceedings of the seventeenth international conference on artificial intelligence and statistics* (pp. 814–822). PMLR. http://proceedings.mlr.press/v33/ranganath14.html

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning* [OCLC: ocm61285753]. MIT Press.

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, *22*(3), 400–407. https://doi.org/10.1214/aoms/1177729586

Robert, C. P., & Casella, G. (2004). *Monte Carlo statistical methods [electronic resource]* [Publication Title: Monte Carlo statistical methods].

Roeder, G., Wu, Y., & Duvenaud, D. (2017). Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference [arXiv: 1703.09194]. *arXiv:1703.09194 [cs, stat]*. Retrieved April 13, 2021, from http://arxiv.org/abs/1703.09194

Sato, M.-a. (2001). Online Model Selection Based on the Variational Bayes. *Neural Computation*, *13*(7), 1649–1681. https://doi.org/10.1162/089976601750265045

Simpson, E., Reece, S., & Roberts, S. J. (2017). Bayesian heatmaps: Probabilistic classification with multiple unreliable information sources. *Joint European conference on machine learning and knowledge discovery in databases*, 109–125.

Spall, J. C. (2003). *Introduction to stochastic search and optimization : Estimation, simulation, and control*. Wiley-Interscience.

Srivastava, A., & Sutton, C. (2017). Autoencoding Variational Inference For Topic Models [arXiv: 1703.01488]. *arXiv:1703.01488 [stat]*. Retrieved April 10, 2021, from http://arxiv.org/abs/1703.01488

Tucker, G., Lawson, D., Gu, S., & Maddison, C. J. (2018). Doubly reparameterized gradient estimators for monte carlo objectives.

Van der Wilk, M., Rasmussen, C. E., & Hensman, J. (2017). Convolutional Gaussian processes. *arXiv preprint arXiv:1709.01894*.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wainwright, M. J., & Jordan, M. I. (2007). Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends® in Machine Learning*, *1*(1–2), 1–305. https://doi.org/10.1561/2200000001

Williams, R. J. (2004). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Yao, Y., Vehtari, A., Simpson, D., & Gelman, A. (2018). Yes, but Did It Work?: Evaluating Variational Inference [ISSN: 2640-3498]. *International Conference on Machine Learning*, 5581–5590. Retrieved February 16, 2021, from http://proceedings.mlr.press/v80/yao18a.html

Zhang, H., Chen, B., Guo, D., & Zhou, M. (2020). WHAI: Weibull Hybrid Autoencoding Inference for Deep Topic Modeling [arXiv: 1803.01328]. *arXiv:1803.01328 [stat]*. Retrieved April 10, 2021, from http://arxiv.org/abs/1803.01328

Department of Engineering Science

Supplementary Questions for 4th Year Project Students

UNIVERSITY OF OXFORD

Factor	Answer	Things to Consider	Record details here
Has the checklist covered all the problems that may arise from working with the VDU?	☑ Yes ☐ No		
Are you free from experiencing any fatigue, stress, discomfort or other symptoms which you attribute to working with the VDU or work environment?	☑ Yes ☐ No	Any aches, pains or sensory loss (tingling or pins and needles) in your neck, back shoulders or upper limbs. Do you experience restricted joint movement, impaired finger movements, grip or other disability, temporary or permanently	
Do you take adequate breaks when working at the VDU?	☑ Yes ☐ No	Periods of two minutes looking away from the screen taken every 20 minutes and longer periods every 2 hours Natural breaks for taking a drink and moving around the office answering the phone etc.	
How many hours per day do you spend working with this computer?	☐ 1-2 ☐ 3-4 ☐ 5-7 ☑ 8 or more		
How many days per week do you spend working with this computer?	☐ 1-2 ☐ 3-5 ☑ 6-7		
Please describe your typical computer usage pattern	I do all my work on it and spend much of my leisure time on it. I use it basically all day, taking regular breaks and using the blue light filter.		

Student Declaration and Academic Approval

Student Declaration:

I have completed the DSE Workstation Checklist and the Supplementary Questions for my computer-related risk assessment for 4YP Project Number indicated below:

4YP Project Number: 12140

4YP Student's Name *(please print)* Benjamin Gutteridge

4YP Student's Signature: Ben Gutteridge

Academic Approval

I confirm my approval of this 4YP DSE Risk Assessment.

Academic Supervisor's Name: *(please print)*

Michael Osborne

Academic Supervisor's Signature:

M Osborne